1-1-1996

Threshold model analysis : computing strategies and application to sire evaluation for calving ease trait

Gamal Abdel Azim
Iowa State University

Threshold model analysis:

Computing strategies and application to sire evaluation for calving ease trait

by

Gamal Abdel Naser Abdel Azim

A thesis submitted to the graduate faculty

in partial fulfillment of the requirements for the degree of

MASTER OF SCIENCE

Department: Animal Science

Major: Animal Breeding

Major Professor: P. Jeffrey Berger

Iowa State University

Ames, Iowa

1996

Graduate College

Iowa State University

This is to certify that the Master's thesis of

Gamal Abdel Naser Abdel Azim

has met the thesis requirements of Iowa State University

Signatures have been redacted for privacy

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

ACKNOWLEDGMENTS

All praise be to God, the cherisher and sustainer of the world, most gracious, most merciful.

I would like to express my thankfulness to my major professor, Dr. Jeffrey Berger for his continuing guidance and support. His constructive criticism and fresh point of view helped make this research much better than it would otherwise have been. The discussion with Dr. Berger was always interesting and fruitful, and without it this work would not have been completed.

Thanks also goes to the members of my graduate committee, Dr. A. E. Freeman, department of Animal Science, and Dr. Paul Hinz, department of Statistics. Their major contribution in my understanding of the methodology in animal breeding and experimental design facilitated this research.

The National Association of Animal Breeders is gratefully acknowledged for providing support for my research assistantship.

I would like to thank many of the friends in the department of Animal Science for our long scientific discussions. These discussions always added to my information and resolved many of the research problems.

I will always be grateful to my parents for their invaluable support throughout all my educational phases.

INTRODUCTION

Sire evaluation for continuous traits with mixed models has received considerable attention from researchers in the last two decades. Sire evaluation based on a mixed model offers the candidates to selection predicted values that are comparable across different populations. This ability of the mixed model procedure to evaluate sires across different environmental and genetic conditions has motivated many scientists to develop mixed model procedures suitable for applications with discontinuous traits.

Threshold model approach has been developed and applied in the last decade as a method of evaluating sires across different populations for discontinuous traits. The threshold model is similar to the mixed linear model because random and fixed effects are incorporated in the model. A threshold mixed model is different from a linear mixed model because a threshold model postulates an underlying continuous variable that it assumed to have a normal distribution.

With categorical data the threshold model procedure has been proved to be superior to the linear mixed model procedure. In many cases the estimates and predictions obtained from a threshold model are always more accurate than the linear mixed model estimates and predictions. The great benefit of a threshold model analysis is impaired by its computing difficulty. Building the equations involves a large number of normal probability integrals. Further the equations need to be set up and solved many times. This makes programs for threshold model estimation difficult to write and test. Few threshold model analysis programs are available and they usually lack generality necessary for a wide range of applications, i.e., the programs are model specific or data specific. Most of these programs are written in FORTRAN77, which limits and confines the programmer to a very small territory of programming tools. This may be one of the reasons the threshold model programs available have been made specific to a limited number of classifications in the model, a limited number of categories in the response variable, a limited number of records in the data, etc.

The objectives of this study are to develop and test a general program for the threshold model analysis, apply the program to dairy sire evaluation for calving ease, and to describe and study the behavior of predictions from a threshold model analysis.

Thesis organization

A separate part for the study conducted on the behavior of predictions from a threshold model analysis is given in a paper format at the end of this thesis.

REVIEW OF LITERATURE

Many traits of interest in livestock production have a discontinuous phenotypic distribution, but are not inherited as a simple Mendelian mechanism. These traits are referred to as threshold traits. Litter size in sheep, calving ease, conformation and type scores, survival or death, and resistance to disease are possible examples for threshold traits. One common feature of these traits is their discrete response, that is, the response is classified rather than measured. For instance, litter size can be classified by two scores, single and twin or more; calving ease can be classified into a number of scores from no problem to extreme difficulty; etc. A threshold character is assumed to have an underlying normally distributed continuous variable with m-1 fixed thresholds delimiting m possible response categories (Falconer, 1989). The underlying variable is inherited in a similar way as any trait measurable on a continuous scale, that is, if the underlying variable were observable, ordinary methods of genetic evaluation of continuous traits could be applicable.

Early studies on the theory and genetic analysis of threshold characters were carried out by Dempster and Lerner (1950). In their study, they defined several properties of the inheritance of threshold characters and described a method to estimate heritability on the outward scale. Using the results of Dempster and Lerner (1950), Gianola (1982) explained further the principals of threshold trait inheritance, described the nature of the additive genetic variance associated with the classes or categories of the phenotypic variable, and also described a method to estimate heritability on the outward scale.

If the outward scale is divided into m observed categories, then the relationship between the phenotype and the underlying variate is taken to be

$$t_{k-1} < y_i \leq t_k$$

for $k \in \{1, 2, \ldots, m\}$, $t_1, t_2, \ldots, t_{m-1}$ are unknown boundary points that partition the continuous scale into m categories. The individual i responds in the k^{th} category if its' realized value y_i belongs to the interval $(t_{k-1}, t_k]$ (Gianola, 1982; Harville and Mee, 1984).

Important genetic properties of threshold traits

1. The effect of a gene substitution in the underlying scale on the genotype is a function of t_k (the k^{th} category), hence, it is not constant throughout the range of the underlying scale. The effect of a gene substitution is computed as the rate of change in θ_k with respect to a^*, where θ_k is the area under the normal curve from k-1 to k, and a^* is the additive genetic value divided by $\sqrt{var(y)}$ (Gianola, 1982).

The effect of a gene substitution = $\frac{\partial \theta_k}{\partial a^*}$ =

$$[2\pi(1-h^2)]^{-1/2}\{\exp^{-(t^*_{k-1}-a^*)^2/2(1-h^2)} - \exp^{-(t^*_k-a^*)^2/2(1-h^2)}\}, \qquad [1]$$

Equation [1] shows that the additive effect of a gene substitution is a function of h^2 (heritability in the underlying scale), t^*_k (the standardized threshold value), and a^* (the standardized additive genetic value).

2. The additive genetic variance of the k^{th} category as derived by Gianola (1982) and Dempster and Lerner (1950) is $(Z_{k-1} - Z_k)^2 h^2$, where Z_{k-1} and Z_k are ordinates of a standardized normal density function corresponding to thresholds between categories k-1 and k. It is obvious from the formulae above that the additive genetic variance corresponding to category k depends on the incidence of that category in the population.

3. Heritability can be estimated for each category of the phenotype. Letting π_k be the probability of response in category k, then $\pi_k(1-\pi_k)$ is the phenotypic variance of such category. From property 2, heritability of the k^{th} category = $(Z_{k-1} - Z_k)^2 h^2 / \pi_k(1-\pi_k)$, and this simplifies to $Z^2 h^2 / \pi(1-\pi)$ with dichotomous traits.

4. Heritability for the whole trait in the outward scale can be computed as

$$h^2[\sum_{k=1}^{m-1} Z_k(n_{k+1} - n_k)]^2 / [\sum_{k=1}^{m} n_k^2 \pi_k - (\sum_{k=1}^{m} n_k \pi_k)^2], \qquad [2]$$

where n_k are weights or scores suggested by Pollak and Freeman (1976) in analyzing calving ease data. $\mathbf{n}' = [n_1\ n_2\ \ldots\ n_m]$ is a vector of scores for m categories.

The above properties might not be of computational use, however, they explain some basic genetic principals of threshold characters. It is important here to emphasize that the

incidence of categories affects the amount of additive genetic variance and heritability of each category. Even if h^2 (heritability on the underlying scale) is the same from one population to another, difference in incidence will bring about different amounts of additive genetic variance (Gianola, 1982). Further, the association of scores in estimating common heritability of the trait may make one set of scores more heritable than another; this property will be used later to prove that the direct analysis of a set of scores without considering an underlying continuous variable is not trustful in predicting correct genetic parameters.

Methods of sire evaluation

Henderson (1973) explained that sires could be selected for a continuous trait according to three types of models:

1. Model I, candidates for selection are fixed factors; sires are dealt with as treatments in an ordinary analysis of variance.
2. Model II, candidates for selection are random samples from a specific population. This model represents the classical methods of selection index.
3. Henderson's mixed model, candidates for selection are randomly sampled from more than one population. The mixed model approach allows the evaluation of sires across different populations.

Linear models have been developed and applied to categorical traits as well. Grizzle et al. (1969) developed a linear fixed model approach to the analysis of categorical data, the method is sometimes referred to as the GSK method(G, S, and K are initials of the authors' last names). The idea behind the method is to fit a number of u functions of unknown true cell probabilities to a linear model, then testing the goodness-of-fit of the model and the significance of any factor in the linear model. The method is appropriate for applications that fit model I, where all factors are fixed. Consider for instance, the problem of fitting a linear model to calving ease data shown in Table 1. The data are collected from 4 herds and 4 sires; three categories of calving ease is assumed.

To explain the method, the following definitions need to be stated.

Define

$\hat{p}_{ijk} = n_{ijk} / n_{ij.}$, where $\hat{p}_{ijk}$ is the estimated cell probability of the i^{th} sire, the j^{th} herd and the k^{th} category; i and j = 1, . . ., 4 and k = 1, 2, 3; if some of the $n_{ijk} = 0$, then $\hat{p}_{ijk}$ can be estimated as $1/rn_{ij.}$, where r is the number of response categories and $n_{ij.}$ is the total number of individuals within a subclass.

Table 1. Frequency distribution of four herds, four sires, and three categories.

Herd															
1				2				3				4			
Sire															
1	2	3	4	1	2	3	4	1	2	3	4	1	2	3	4
n_{111}[1]	n_{121}	n_{131}	n_{141}	n_{211}	. .	. .	. .	n_{311}	. .	. .	. .	n_{411}	. .	. .	n_{441}
n_{112}	n_{122}	n_{132}	n_{142}	. .	. .	. .	. .	. .	. .	. .	. .	. .	. .	. .	. .
n_{113}	n_{123}	n_{133}	n_{143}	n_{213}	. .	. .	. .	n_{313}	. .	. .	. .	n_{413}	. .	. .	n_{443}
$n_{11.}$	$n_{12.}$	$n_{13.}$	$n_{14.}$	$n_{21.}$	. .	. .	. .	$n_{31.}$	. .	. .	. .	$n_{41.}$	. .	. .	$n_{44.}$

[1] n_{ijk} denotes the number of observations in the i^{th} herd, j^{th} sire, and k^{th} category.

$\hat{\mathbf{p}}'_{ij} = [\hat{p}_{ij1}\ \hat{p}_{ij2}\ \hat{p}_{ij3}]$; $\hat{\mathbf{p}}'_{1x48} = [\hat{\mathbf{p}}'_{11}\ \hat{\mathbf{p}}'_{12} \ldots \hat{\mathbf{p}}'_{44}]$;

$$\text{var}(\hat{\mathbf{p}}'_{ij}) = \frac{1}{n_{ij.}} \begin{bmatrix} p_{ij1}(1-p_{ij1}) & -p_{ij1}p_{ij2} & -p_{ij1}p_{ij3} \\ & p_{ij2}(1-p_{ij2}) & -p_{ij2}p_{ij3} \\ \text{sym.} & & p_{ij3}(1-p_{ij3}) \end{bmatrix}; \text{ and}$$

$\sum_{\hat{p}}$ = 48x48 block diagonal matrix having $\text{var}(\hat{\mathbf{p}}'_{ij})$ on the main diagonal.

The model of interest is composed of additive factors(herd and sire), contributing to the probability of falling in one of the three categories. The following model formulates the contribution of the i^{th} herd (i = 1, . . ., 4) and the j^{th} sire (j = 1, . . ., 4) to the probability of falling in the k^{th} category

$$\hat{p}_{ijk} = \mu + \alpha_i + \tau_j + \varepsilon_{ij.} \qquad [3]$$

The following procedures examine the adequacy of the model [3] or the goodness-of-fit of the model to the data; the model fits the data if the null hypothesis, Ho: $\varepsilon_{ij} = 0$, proves true. The mean score for each sire within each herd calculated by $1p_{i1} + 2p_{i2} + 3p_{i3}$ is taken to represent the probability of interest. Choose the following matrix, **A**, which will be used to formulate the mean values

$$\mathbf{A}_{16x48} = \begin{bmatrix} 1 & 2 & 3 & 0 & 0 & 0 & 0 & 0 & 0 & \cdots & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 2 & 3 & 0 & 0 & 0 & \cdots & 0 & 0 & 0 \\ \vdots & & & & & & & & & & & & \\ \vdots & & & & & & & & & & & & \\ \vdots & & & & & & & & & & & & \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & \cdots & 1 & 2 & 3 \end{bmatrix};$$

then set $\mathbf{Ap} = \mathbf{X}\beta$, where $\mathbf{X}$ is a known incidence matrix of an additive model having herd and sire effects. The $\mathbf{X}$ relates subclasses to levels in the model, in our case the dimension of $\mathbf{X}$ is 16 x 9. The β is a 9x1 vector of parameters, i.e., $\beta' = [\mu\ \alpha_1\ \alpha_2\ \alpha_3\ \alpha_4\ \tau_1\ \tau_2\ \tau_3\ \tau_4]$. Denote $\mathbf{Ap} = \mathbf{y}$ and $\text{var}(\mathbf{Ap}) = \mathbf{A}\,\Sigma_{\hat{p}}\,\mathbf{A}' = \Sigma_Y$. The β can then be estimated by generalized least squares as

$$\beta = (\mathbf{X}'\Sigma_Y^{-1}\mathbf{X})^{-}\mathbf{X}'\Sigma_Y^{-1}\mathbf{y}. \qquad [4]$$

Estimability conditions of linear functions of $\hat{\beta}$ should be the same as in any general least square estimation (Searle, 1971). Further, $(\mathbf{X}'\Sigma_Y^{-1}\mathbf{X})^{-}$ is a generalized inverse to which $\hat{\beta}$ is invariant (Gianola, 1982).

Consider now testing the goodness-of-fit of the model and testing hypotheses about the parameters in the linear model as well. Grizzle et al. (1969) suggested the following statistic to test the goodness-of-fit

$$\mathbf{y}'\Sigma_Y^{-1}\mathbf{y} - \hat{\beta}'\mathbf{X}'\Sigma_Y^{-1}\mathbf{X}\hat{\beta}. \qquad [5]$$

Equation [5] follows a χ^2 distribution with 16 - 9 degrees of freedom.

If the model fits the data, any factor in the model can be tested as well. For instance, to test α_i or herd effect, select a matrix

$$\mathbf{C} = \begin{bmatrix} 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \end{bmatrix},$$

to test H_o : $\mathbf{C}\beta = \mathbf{0}$ with $\mathbf{C}$ of rank r (r = 4 in our case). The suggested test statistic in the context of the GSK method is

$$(\mathbf{C}\hat{\beta})'[\mathbf{C}(\mathbf{X}'\Sigma_Y^{-1}\mathbf{X})^{-}\mathbf{C}']^{-1}\mathbf{C}\hat{\beta}. \qquad [6]$$

Equation [6] also follows a χ^2 distribution with r degrees of freedom.

Based on the GSK method Schaeffer and Wilton (1976) developed a method of sire evaluation for calving ease and livability of calf as a combined threshold trait classified into 6 categories:

1 alive, normal; 2 alive, slight difficulty; 3 alive, extreme difficulty; 4 dead, normal; 5 dead, slight difficulty; and 6 dead, extreme difficulty.

Schaeffer and Wilton reported that the GSK definition of a population is not suitable for sire evaluation of calving ease because the number of observations in real data do not allow for adequate estimation of the cell probabilities, $\hat{p}_{ijk}$, which are required to compute Σ_Y^{-1}. Hence, they suggested assuming only one population and computed $\hat{p}_k$ as

$$\hat{p}_k = \Sigma\, i \in R_k / N,$$

where N = the total number of observations, and $\Sigma\, i \in R_k$ = the sum of all individuals pertaining to the set R_k, i.e., all individuals of category k.

Define

$$\hat{\mathbf{p}}' = \begin{bmatrix} \hat{p}_1 & \hat{p}_2 & \hat{p}_3 & \hat{p}_4 & \hat{p}_5 & \hat{p}_6 \end{bmatrix};$$

$$\text{var}(\hat{\mathbf{p}}) = \frac{1}{N} \begin{bmatrix} \hat{p}_1(1-\hat{p}_1) & -\hat{p}_1\hat{p}_2 & \cdots & -\hat{p}_1\hat{p}_6 \\ & \hat{p}_2(1-\hat{p}_2) & \cdots & -\hat{p}_2\hat{p}_6 \\ & & \ddots & \vdots \\ & & & \vdots \\ \text{sym.} & & & \hat{p}_6(1-\hat{p}_6) \end{bmatrix};$$

and

$$\Sigma\hat{\mathbf{p}} = \frac{1}{n_i} \overset{s}{\underset{i=1}{\Sigma^+}} \text{var}(\hat{\mathbf{p}}),$$

where Σ^+ is the direct sum. This gives a block diagonal matrix of order rxs, where r and s indicate the number of categories and subclasses, respectively. Notice that the matrix $\Sigma\hat{\mathbf{p}}$ has $1/n_i$ var$(\hat{\mathbf{p}})$ on the main diagonal. The division by n_i accounts for the number of observations in each subclass.

The assumption of one population was followed by Schaeffer an Wilton to compute only $\Sigma_{\mathbf{Y}}$,($\Sigma_{\mathbf{Y}} = \mathbf{A}\sum\hat{\mathbf{p}}\,\mathbf{A}'$), however, the estimates of cell probabilities from each subclass were computed as before in the GSK method, these estimates were used to compute $\mathbf{y}$ ($\mathbf{y} = \mathbf{A}\hat{\mathbf{p}}$), and not $\Sigma_{\mathbf{Y}}$.

The model equations used here are similar to that of the GSK method, the equations are: $\mathbf{y} = \mathbf{X}\beta$. However, they can be solved easier here since var($\hat{\mathbf{p}}$) is identical for all the diagonal elements of $\sum\hat{\mathbf{p}}$. Thus, $\hat{\beta} = (\mathbf{X'DX})^{-}\mathbf{X'Dy}$, where $\mathbf{D}$ represents the diagonal matrix whose elements are n_i.

After solving for $\hat{\beta}$, fixed effects including sires are then ranked according to $\hat{\beta}$. This approach of ranking fixed sires restricts any inference about them to the conditions of the experiment, besides no inference can be made about the population of these sires.

Incorporating random effects

The inadequacy of ranking sires when considered as fixed factors in the model has led many researchers like Pollak and Freeman (1976) and Berger and Freeman (1978) to try to incorporate random sires in the model. Pollak and Freeman (1976) applied the mixed model approach (Henderson 1973) to sire evaluation for dystocia. Sires were sorted according to their random solutions. Both random and fixed solutions were obtained by solving iteratively the following linear system of mixed model equations

$$\begin{bmatrix} \mathbf{X'X} & \mathbf{X'Z} \\ \mathbf{Z'X} & \mathbf{Z'Z}+\mathbf{D}^{-1} \end{bmatrix}\begin{bmatrix} \hat{\beta} \\ \hat{\mathbf{u}} \end{bmatrix} = \begin{bmatrix} \mathbf{X'y} \\ \mathbf{Z'y} \end{bmatrix}, \qquad [7]$$

where $\mathbf{X}$ and $\mathbf{Z}$ are known design matrices, $\hat{\beta}$ and $\hat{\mathbf{u}}$ are two vectors of estimated fixed and random parameters respectively, $\mathbf{D}^{-1} = \alpha\mathbf{A}^{-1}$, where $\alpha = \sigma_e^2 / \sigma_u^2$ and $\mathbf{A}$ is Wright's numerator relationship matrix, and $\mathbf{y}$ is a vector of categorical observations.

Berger and Freeman (1978) presented a similar approach which adjusts for the unequal error variance associated with the parity-of-dam effect. They partitioned the residual variance covariance matrix into several mutually heterogeneous submatrices. It is illustrative to

mention that in the categorical data the variance in the outward scale depends on the distribution of the categories i.e., the phenotypic variance of, for instance, a dichotomous trait distributed according to a binomial distribution is obtained by $\theta_j(1 - \theta_j)$, where θ_j is the incidence of the trait in the j^{th} subpopulation. Since θ_j is different across different subpopulations, one expects heterogeneous variance across those subpopulations.

Problems associated with sire evaluation for categorical traits using Henderson's mixed models

Schaeffer and Wilton (1976) reported that BLUP is applicable to discrete data as well as continuous data because it does not assume any distribution. Thompson (1979), however, discouraged the idea of applying linear models to dichotomous traits and proposed another procedure which has been adopted by scientists in developing threshold models.

Several studies have indicated that BLUP is not appropriate unless the response variable is quantitative and follows a fixed, mixed or random linear statistical model(Gianola, 1980; Gianola, 1982; Gianola and Foulley, 1983; Harville and Mee, 1984). Gianola (1980) discussed several problems associated with applying BLUP directly to categorical scores. First, assigning scores to response categories is arbitrary and one set of scores can give different heritability estimate than another set; it was mentioned before that heritability is dependent on the assigned scores (property 4, [2]). This implies that BLUP should be applied to the underlying continuous response variable rather than to arbitrary scores. This also complies with the additivity assumptions implicit in the mixed model (Harville and Mee, 1984). In other words, the additive model may not fit categorical scores whereas it fits an underlying continuous variable.

Second, in applying mixed model estimation to categories, the restriction that the sum of response probabilities must equal one is not taken into consideration. This implies that the method does not exclude unfeasible predicted probabilities of values less than 0 or greater than 1.

Third, a considerable amount of nonadditive genetic variance is associated with the observed scale. Dempster and Lerner (1950) showed that the contribution of the nonadditive

genetic variance for a binary trait in the outward scale increases as the incidence becomes more extreme. This indicates that by using observed scores, it is unlikely to get estimates of the additive genetic variance free of dominance or epistatic biases. Therefore, mixed model solutions would give predictors with unnecessarily large sampling variances.

Finally, categories are always assigned numbers 1, 2, 3, . . ., m, which is important in ordering categories but misleading in its assumption of equal intervals between categories. The analysis based on the threshold concept avoids needing to make this unwarranted assumption (Bock, 1975).

Scaling ordered categorical data

Snell (1964) presented a method of scaling ordered categorical data. The method assumes the presence of an underlying continuous variable which follows a normal distribution. The underlying normal distribution is approximated with a logistic distribution. Another approximation replaces theoretical probabilities in the first derivative of the log-likelihood function by observed probabilities to solve for the values of boundary points.

Due to the fact that Snell's method makes the residual variance homogeneous and normally distributed, many researchers have applied the method to dystocia data (e.g.,Tong et al., 1977; Naazie, et al., 1989; Naazie et al., 1991).

The method has two shortcomings, first, it involves several approximations such as approximating the normal distribution with a logistic distribution and replacing theoretical probabilities by observed probabilities. Second, with small numbers of observations, the approximation has been proven to be inadequate (Snell, 1964).

The threshold approach

Early studies of the threshold approach were carried out by McKelvey and Zavoina (1975). They gave a complete description of a fixed-effects threshold model. The fixed version of the threshold model has been extended by Harville and Mee (1984) to include random effects as well.

Following the threshold concept, an underlying continuous variable **y** is assumed to follow a linear mixed model. The model for **y** is

$$\mathbf{y} = \mathbf{X}\alpha + \mathbf{U}\beta + \mathbf{e}, \qquad [8]$$

where $\mathbf{y}' = [Y_1\ Y_2\ \ldots\ Y_n]$, **X** and **U** are known incidence matrices, α and β are fixed and random unknown parameters, respectively, and **e** is a random residual. This model keeps all the assumptions of a mixed model. Further, it is assumed that **y** is partitioned by m+1 boundary points into m intervals representing phenotypic categories. The phenotype of an individual i = k if

$$\xi_{k-1} < Y_i \leq \xi_k, \qquad [9]$$

where k = 1, 2, . . ., m; i = 1, 2, . . ., n; and ξ_k is the k^{th} boundary point. It is assumed that $\xi_0 = -\infty$, $\xi_m = +\infty$, and $\xi_{k-1} \leq \xi_k \leq \ldots \xi_m$. The phenotypic variable will be denoted as **z** and the phenotype of the i^{th} individual by Z_i.

Many researchers reported that the distribution of **y** can be assumed to be normal (Dempster and Lerner, 1950; Bulmer, 1980; Gianola, 1982; Gianola and Foulley, 1983). Hence, the density function of **y** can be considered to be the normal distribution function and the probability function of the observed dependent variable can be written as

$$\Pr[Z_i \in R_k] = \Phi(\xi_k - \mu_i) - \Phi(\xi_{k-1} - \mu_i), \qquad [10]$$

where R_k is the set of individuals responding in category k; Φ represents the cumulative standard normal distribution function, i.e.,

$\Phi(t) = \int_{-\infty}^{t} \frac{1}{\sqrt{2\pi}} e^{-X^2/2} dX$, notice that the threshold model can be reformulated into another model with an error variance = $\mathbf{I}\sigma_e^2$, where $\sigma_e^2 = 1$ (Harville and Mee, 1984);

$\mu_i = \mathbf{x}_i'\alpha + \mathbf{u}_i'\beta$, where $\mathbf{x}_i'$ and $\mathbf{u}_i'$ represent the i^{th} row of **X** and **U**, respectively and α and β are unknown parameters. Maximum likelihood methods can then be used to obtain estimates of the population parameters, ξ, α, and β, of the model [10].

If **y** were observable we could maximize the joint distribution of **y** and β to arrive at the mixed model equations, i.e. maximizing

$$\Phi\left(\begin{bmatrix}\mathbf{y}\\ \boldsymbol{\beta}\end{bmatrix};\begin{bmatrix}\mathbf{X}\boldsymbol{\alpha}\\ \mathbf{0}\end{bmatrix},\begin{bmatrix}\mathbf{V} & \mathbf{UD}\\ \mathbf{DU}' & \mathbf{D}\end{bmatrix}\right) = \Phi(\mathbf{y};\mathbf{X}\boldsymbol{\alpha}+\mathbf{U}\boldsymbol{\beta},\mathbf{I})\,\Phi(\boldsymbol{\beta};\mathbf{0},\mathbf{D})$$

$$= \Phi(\boldsymbol{\beta};\mathbf{0},\mathbf{D})\prod_{i=1}^{n}\phi(Y_i-\mu_i), \qquad [11]$$

where $\mu_i = \mathbf{x}_i'\boldsymbol{\alpha}+\mathbf{u}_i'\boldsymbol{\beta}$, $\mathbf{V}$ = var($\mathbf{y}$), $\mathbf{D}$ = var($\boldsymbol{\beta}$), ϕ denotes the univariate standard normal distribution function, and Φ denotes the multivariate normal distribution function. Harville and Mee (1984) used a similar approach to derive the threshold model equations making use of results from Bock (1975), McKelvey and Zavonia (1975), and Mee (1981). Define

$$\Theta(\mathbf{z};\,\xi,\,\alpha,\,\beta) = \prod_{i=1}^{n}\int_{\xi_{z_{i-1}}}^{\xi_{z_i}}\phi(Y_i-\mu_i)\,dY_i. \qquad [12]$$

Equation [12] shows that the interest is not in a distribution of points, $\phi(Y_i-\mu_i)$, as in the derivation of the mixed model equations, but rather it is in a distribution of areas under the normal curve, $\int_{\xi_{z_{i-1}}}^{\xi_{z_i}}\phi(Y_i-\mu_i)\,dY_i$. Now replace equation [12] in equation [11] and maximize

$$\Phi(\boldsymbol{\beta};\mathbf{0},\mathbf{D})\prod_{i=1}^{n}\int_{\xi_{z_{i-1}}}^{\xi_{z_i}}\phi(Y_i-\mu_i)\,dY_i, \qquad [13]$$

Because maximizing equation [13] is equivalent to maximizing its natural log, then let L denote the natural log of [13] and maximize L

$$L = \sum_{i=1}^{n}\left\{\ln\left[\Phi(\xi_k-\mu_i)-\Phi(\xi_{k-1}-\mu_i)\right]\right\} + C - \frac{1}{2}\boldsymbol{\beta}'\mathbf{D}^{-1}\boldsymbol{\beta}, \qquad [14]$$

where $\int_{\xi_{z_{i-1}}}^{\xi_{z_i}}\phi(Y_i-\mu_i)\,dY_i$ can be evaluated as $\Phi(\xi_k - \mu_i) - \Phi(\xi_{k-1} - \mu_i)$, and $C = \ln\left(\frac{1}{(2\pi)^{q/2}|\mathbf{D}|^{1/2}}\right)$. The first derivative of L is computed with respect to the three unknowns, ξ, α, and β and then equated to $\mathbf{0}$ to get a system of equations which gives estimators for the unknowns after being solved iteratively. Define

$$\frac{\partial L}{\partial \xi} = \nu(\xi,\,\alpha,\,\beta;\,\mathbf{z}); \qquad [15]$$

$$\frac{\partial L}{\partial \alpha} = \mathbf{X}'\varepsilon(\xi, \alpha, \beta; \mathbf{z}); \qquad [16]$$

$$\frac{\partial L}{\partial \beta} = \mathbf{U}'\varepsilon(\xi, \alpha, \beta; \mathbf{z}) - \mathbf{D}^{-1}\beta; \qquad [17]$$

and equate the first partials computed from [15], [16], and [17] to **0**. This yields the following system of equations

$$\begin{bmatrix} \nu(\xi, \alpha, \beta; \mathbf{z}) \\ \mathbf{X}'\varepsilon(\xi, \alpha, \beta; \mathbf{z}) \\ \mathbf{U}'\varepsilon(\xi, \alpha, \beta; \mathbf{z}) - \mathbf{D}^{-1}\beta \end{bmatrix} = \mathbf{0}. \qquad [18]$$

To describe the elements of the above system of equations, define

$\phi_{ik} = \phi(\xi_k - \mu_i)$;

$\Phi_{ik} = \Phi(\xi_k - \mu_i)$;

Then, $\nu(\xi, \alpha, \beta; \mathbf{z})$ can be described as (m-2) x 1 vector whose $(k-1)^{th}$ element is

$$\Sigma_{i \in R_k}\left(\frac{\phi_{ik}}{\Phi_{ik} - \Phi_{i,k-1}}\right) - \Sigma_{i \in R_{k-1}}\left(\frac{\phi_{ik}}{\Phi_{i,k+1} - \Phi_{ik}}\right), \qquad [19]$$

notice that the length of $\nu(\xi, \alpha, \beta; \mathbf{z})$ = m-2 because $\xi_1 = 0$ in the standardized threshold model(Harville and Mee, 1984); also, remember that $\xi_0 = -\infty$ and $\xi_m = +\infty$.

ε is an n x 1 vector whose i^{th} element is

$$\varepsilon_i = \frac{\phi_{i,Z_i-1} - \phi_{i,Z_i}}{\Phi_{i,Z_i} - \Phi_{i,Z_i-1}}, \qquad [20]$$

Z_i = the category into which the i^{th} individual responds, if the i^{th} individual responds in the category for instance, k, then $Z_i = k$ and $Z_{i-1} = k-1$.

The next step is to compute $\hat{\xi}$, $\hat{\alpha}$, and $\hat{\beta}$ which are solutions of the system of equations, [18]. Unfortunately, the equations [18] are not linear in the unknowns, hence, they cannot be solved by ordinary methods of simultaneous linear equations. McKelvey and Zavoina (1975) suggested the Newton Raphson method to get solutions for $\hat{\xi}$ and $\hat{\alpha}$ in their fixed threshold model; Harville and Mee (1984) followed a similar approach to get solutions

for $\hat{\xi}$, $\hat{\alpha}$, and $\hat{\beta}$ in a mixed threshold model. The algorithm of Harville and Mee is shown below

$$\begin{bmatrix} \mathbf{Q} & \mathbf{L}'\mathbf{X} & \mathbf{L}'\mathbf{U} \\ \mathbf{X}'\mathbf{L} & \mathbf{X}'\mathbf{R}\mathbf{X} & \mathbf{X}'\mathbf{R}\mathbf{U} \\ \mathbf{U}'\mathbf{L} & \mathbf{U}'\mathbf{R}\mathbf{X} & \mathbf{U}'\mathbf{R}\mathbf{U}+\mathbf{D}^{-1} \end{bmatrix} \begin{bmatrix} \hat{\xi} \\ \hat{\alpha} \\ \hat{\beta} \end{bmatrix} = \begin{bmatrix} \nu(\xi,\alpha,\beta;\mathbf{z}) \\ \mathbf{X}'\varepsilon(\xi,\alpha,\beta;\mathbf{z}) \\ \mathbf{U}'\varepsilon(\xi,\alpha,\beta;\mathbf{z})-\mathbf{D}^{-1}\beta \end{bmatrix}, \qquad [21]$$

where the right hand side is defined as before and the matrix **D** is defined as in [7]. The matrices **Q**, **L**, **R** are defined as follows:

L is the n x (m-2) matrix whose (i, k-1)th element is

$$\phi_{ik}\{(\delta_{ik}/\Delta_{ik})-(\delta_{i,k+1}/\Delta_{i,k+1})\}, \qquad [22]$$

where $\delta_{ik} = \phi_{i,k-1} - \phi_{ik}$ and $\Delta_{ik} = \Phi_{ik} - \Phi_{i,k-1}$;

Q is the (m-2) x (m-2) tridiagonal matrix whose (j-1, k-1)th element is

$$\begin{cases} \sum_{i=1}^{n}\phi_{ij}^2(\Delta_{ij}^{-1}+\Delta_{i,j+1}^{-1}) & \text{if } j = k, \\ -\sum_{i=1}^{n}\phi_{ij}\phi_{ik}/\Delta_{ik} & \text{if } j = k-1, \\ -\sum_{i=1}^{n}\phi_{ij}\phi_{ik}/\Delta_{ik} & \text{if } j = k+1, \\ 0 & \text{otherwise;} \end{cases} \qquad [23]$$

and **R** is nxn diagonal matrix whose ith diagonal element is

$$\sum_{k=1}^{m}\delta_{ik}^2/\Delta_{ik}. \qquad [24]$$

Gianola and Foulley (1983) used a Bayesian approach to derive an algorithm similar to [21]. They presented both normal and logistic cases. In the normal case, they considered the conditional distribution of **y** to be normal. In the logistic case, they approximated the normal integrals with a logistic function, that is,

$\Pr[Z_i \in R_k] = \Phi_{ik} - \Phi_{i,k-1} \cong C_{ik} - C_{i,k-1}$, where

$$C_{ik} = \left[1+e^{-\frac{\pi}{\sqrt{3}}(\xi_k-\mu_i)}\right]^{-1}.$$

The normal case leads to equivalent estimation equations to those proposed by Harville and Mee (1984) and presented earlier. Further, the approach they used is consistent with the extension of generalized linear models for a binary variable suggested by Thompson (1979) (Gianola and Foulley, 1983).

Computer programs available for threshold model analysis

Early attempts to develop programming strategies for building and solving threshold model equations have been carried out by Djemali (1985). A computer program was developed to analyze calving ease data. The program is designed for a specific model and is restricted to five response categories. Misztal et. al. (1989) developed a computing strategy for computing estimates and predictions in a threshold mixed model. A computer program (CMMAT program) was written in FORTRAN77. The program handles a limited number of factors in the model and a limited number of response categories. Further, the structure of the equations is built in one way to get sire evaluations for a threshold trait. The equations cannot be altered by the user to any other structure, e.g., the equations can not be modified from a sire model to an animal model. Finally, the CMMAT program uses approximate tabulated values for the normal distribution and integral functions involved in the threshold model computations.

Threshold model estimation for dystocia

The degree of difficulty associated with the birth of a calf is classified into a number of classes beginning with easy calving and ending with extreme difficulty. In dairy production systems dairy producers assign 5 scores to calving difficulty as follows:
1 (no problem); 2 (slight difficulty); 3 (needed assistance); 4 (considerable force needed); 5 (extreme difficulty).

Sire evaluation for calving ease has passed through several stages, early studies have been carried out by Schaeffer and Wilton (1976). They ranked sires in a completely fixed model. Pollak and Freeman (1976) and Berger and Freeman (1978) applied the mixed model procedure to evaluate sire merits for calving ease. In the United States calving ease scores

were analyzed by the mixed sire model procedure until 1987. An ordered categorical analysis by a threshold model has been adopted since 1988, (Clutter et al., 1989; Djemali and Berger, 1987).

A model for dystocia analysis

Several factors other than sire of the calf have a significant impact on dystocia and have to be included in the model. The following are the most prominent factors:

1. Parity-of-dam

Philipsson (1976a) reported 15.7% of dystocia in heifers versus 4.8% in cows. Berger (1994) reported that fewer cows needed assistance for calving in second, third and later parities than first parity in data collected over 14 years, (1978 - 1992). Therefore, adjustment for parity is very important in the analysis of dystocia. A common practice in analyzing dystocia scores is to classify parities into 3 levels within the parity factor; first-parity, second-parity, and third-or-subsequent parities.

2. Sex-of-calf

Due to the bigger size of male calves, higher dystocia rates are to be expected with male than female calves. Pollak and Freeman (1976) and Berger and Freeman (1978) pointed out that sex-of-calf was a very important source of variation for dystocia.

3. Herd-Year-Season(HYS)

Dystocia is affected by season of calving. Pollak and Freeman (1976) reported that winter births were more difficult than summer births. Summer is a grazing season for cows, that allows them to exercise while on pasture which leads to less dystocia rates in summer seasons than winter seasons (Philipsson, 1976b).

If only one of the categories could be found in a HYS subclass a major problem is encountered in maximizing [13] as it will be strictly increasing or decreasing in that case. This may lead to a computational problem in solving the computing algorithm, [21] (Harville and Mee, 1984). Two alternatives may be used to avoid this problem, deleting such HYS subclasses or considering HYS as a random effect in the threshold model. Many researchers

favored the second option as deleting subclasses could lead to a loss of a substantial amount of information (e.g., Djemali and Berger, 1987; Clutter et al., 1989).

MATERIAL AND METHODS

The objective of this chapter is to describe procedural steps for developing a computer program to build the threshold model equations, to validate the program using simulated data, and to explain how to use the program to analyze calving ease data in the context of the threshold model.

Threshold mixed model analysis program

A computer program was written to build the matrices of the threshold part in the threshold model equations. The program was designed to be flexible enough to accommodate any number of fixed or random factors and any number of categories. Further the program is not model specific and can handle any structure of the threshold model equations.

The objective of the program is to compute solutions for the unknowns in the threshold model equations, [21]. The threshold model equations are different from the conventional mixed model equations in that solutions of the $(k-1)^{th}$ iterate are used to build the equations for the k^{th} time. More specifically the matrices **Q, L, R**, and the vectors ν and ε, which will be referred to as the matrices of the threshold part, of the k^{th} iterate are all functions of the solutions of the $(k-1)^{th}$ iterate, hence they have to be built all over again every iterate. The matrices **X**, **Z**, and **D,** which will be referred to as the matrices of the mixed part, are not functions of the solutions of the previous iterate, therefore, they are built only once.

Programming language

The C programming language was used to write the threshold model analysis program (will be referred to as TMA), (Curry, 1991; Oualline, 1993; Press et. al., 1992a; Press et. al., 1992b; Schildt, 1990). C was selected because it allows more flexible and efficient programming than other programming languages, especially if the programs will be used to handle big data sets. For example, pointers are one of the features of C that makes it superior to many other programming languages. A pointer is defined as a variable that holds the memory address of another object. Pointers can be used to manipulate arrays more efficiently by moving pointers to them instead of moving the arrays themselves, this advantage is very

important in terms of time efficiency if a program is written to handle arrays of large size. In addition, pointers are used to communicate information about memory. This allows defining arrays with unlimited dimensions, i.e., the dimensions of the arrays are to be known to the program during the run time depending on the size of the data. Of course, this is better than the old common programming practice, (e.g., in FORTRAN77) of assigning dimensions to any defined array. By assigning dimensions to arrays, the program will consume a fixed amount of memory based on the assigned dimensions no matter how big or small the amount of data. Another problem associated with the fixed array size occurred when using more data than the arrays of the program can hold, in this case a modification of the code followed by recompiling the program will be required.

Building the equations for the first time

Initializing the iterative algorithm

The threshold model equations need to be built and solved several times. The equations are solved each time for the difference between two successive sets of solutions. Define the vector of solutions, the right hand side of the equations, the left hand side of the equations of the k^{th} iterate as $\mathbf{S}^{(k)}$, $\mathbf{LHS}^{(k)}$, $\mathbf{RHS}^{(k)}$, respectively. Then solutions of the $(k+1)^{th}$ round are computed in two steps. First, solve the system of equations, $\mathbf{LHS}^{(k)}[\ \mathbf{S}^{(k+1)} - \mathbf{S}^{(k)}] = \mathbf{RHS}^{(k)}$, for the corrections, $\mathbf{S}^{(k+1)} - \mathbf{S}^{(k)}$. Second, add the corrections to the k^{th} round to get the set of solutions of the $(k+1)^{th}$ round, $\mathbf{S}^{(k+1)}$. This process is repeated until the corrections become zero.

To build the equations for the first time initial solutions are needed. The program was designed to compute starting values, $\mathbf{S}^{(0)}$, to initiate the algorithm. Notice that the solution vector, $\mathbf{S}^{(0)}$ represents estimates of m-2 unknown boundary points, fixed effects, and random effects respectively, i.e., $\mathbf{S}^{(0)} = [\hat{\xi}^{(0)}\hat{\alpha}^{(0)}\hat{\beta}^{(0)}]'$. Let $\Phi(t_k) = N_k/N$, where N_k is the total number of individuals in categories 1 to k, and N is the total number of individuals. The values of t_k were then computed using a function which computes the area from $-\infty$ to t_k under the standard normal curve and computes the point t_k. The following rational fraction was used to program the function, (Kennedy, 1980)

$$t \cong \tau - (\rho_0 + \rho_1\tau + \rho_2\tau^2) / (1 + \lambda_1\tau + \lambda_2\tau^2 + \lambda_3\tau^3),$$

where $t_k = t$; the area from $-\infty$ to $t = p$; $\tau = \sqrt{-2\log(1-p)}$, $0.5 \leq p < 1$; and

$\rho_0 = 2.515517$ $\lambda_1 = 1.432788$

$\rho_1 = 0.802853$ $\lambda_2 = 0.189269$

$\rho_2 = 0.010328$ $\lambda_3 = 0.001308$.

Based on the computed values of t_k, starting set of solutions can be computed by setting

$\hat{\xi}_1^{(0)}$ corresponding to the first unknown boundary point to 0.0,

$\hat{\xi}_k^{(0)}$ corresponding to the 2nd until the (m-1)th unknown boundary point to $t_k - t_1$,

$\hat{\alpha}_1^{(0)}$ corresponding to the overall mean to $- t_1$,

$\hat{\beta}^{(0)}$ corresponding to the vector of the unknown random parameters to **0.**

In general, the solutions are not used directly to build the matrices of the threshold part, rather they are used to compute the parameters, ϕ_{ik}, Φ_{ik}, δ_{ik}, and Δ_{ik}, these parameters are used to build the matrices of the threshold part. The following explains the procedure used to compute the parameters, ϕ_{ik}, Φ_{ik}, δ_{ik}, and Δ_{ik}:

a. $\phi_{ik} = \phi(\xi_k - \mathbf{x}_i'\alpha - \mathbf{u}_i'\beta)$, where

$\phi(x)$ is a function used to compute the value of the dependent variable of the normal density function at point x of the independent variable. If $\xi_k = t_k - t_1$ and $\alpha_1 = -t_1$, then $\phi_{ik} = \phi((t_k - t_1) + t_1) = \phi(t_k)$. It is obvious that $\phi(t_k)$ can be used for any individual, i, however, this is only true for building the equations for the first time. Notice that $\phi_{i0} = \phi_{im} = \phi(\pm\infty) = 0.0$ and $\phi_{i1} = \phi(0.0 - \alpha_1) = \phi(t_1)$.

b. $\Phi_{ik} = \Phi(\xi_k - \mathbf{x}_i'\alpha - \mathbf{u}_i'\beta)$, where

$\Phi(x)$ is a numerical integration function that uses the upper limit of integration, x, and computes the area under the standard normal curve from negative infinity to the point x. The routine s15abf in the NAG library was used to compute $\Phi(x)$, (NAG, 1993). In building the equations for the first time $\Phi_{ik} = \Phi(t_k)$, and, in general, $\Phi_{i0} = 0.0$ and $\Phi_{im} = 1.0$

c. $\delta_{ik} = \phi_{i,k-1} - \phi_{ik}$, and

d. $\Delta_{ik} = \Phi_{ik} - \Phi_{i,k-1}$.

Building **L**

The matrix **L** is defined in [22] as an n x (m-2) matrix whose (i, k-1)th element is $\phi_{ik}\{(\delta_{ik} / \Delta_{ik}) - (\delta_{i,k+1} / \Delta_{i,k+1})\}$.

The procedures followed to build **L** for the first time were simpler in the first round than in the second and later rounds because ϕ_{ik}, δ_{ik}, Δ_{ik}, $\delta_{i,k+1}$, and $\Delta_{i,k+1}$ are computed once and used for any individual (any element of **L** is not a function of the ith row of **X** or **Z**). This is also true for all of matrices and vectors in the threshold part of the equations, except for ε.

The program does not generate **L** if there are only two categories detected in the categorical observations(e.g., binomial variable). Also, **L** reduces to a vector if there are only three categories found in the data.

Building **Q:**

The matrix **Q** is defined in [23] as an (m-2) x (m-2) tridiagonal matrix whose (j-1, k-1)th element can be one of the following:

1. $\Sigma_{i=1}^{n}\phi_{ij}^{2}(\Delta_{ij}^{-1} + \Delta_{i,j+1}^{-1})$ if j = k, or for any diagonal element,

2. $-\Sigma_{i=1}^{n}\phi_{ij}\phi_{ik} / \Delta_{ik}$ if j = k-1, or for any upper diagonal element,

3. $-\Sigma_{i=1}^{n}\phi_{ij}\phi_{ik} / \Delta_{ij}$ if j = k +1, or for any lower diagonal element, or

0.0 otherwise.

In building **Q** for the first time $\Sigma_{i=1}^{n}$ is replaced by multiplying by n, expressions 1, 2, and 3 are the same for any i. Further **Q** is symmetric, therefore only 1 and 2 need to be computed.

The matrix **Q** is not produced with a dichotomous response variable and reduces to a 1x1 matrix with trichotomous response variable. Figure 1 shows the symmetric tridiagonal structure of $\mathbf{Q}_{3x3}$ with five categories in the response variable.

$$\begin{bmatrix} n\phi_1^2(\Delta_1^{-1} + \Delta_2^{-1}) & -n\phi_2\phi_3 / \Delta_3 & 0.0 \\ & n\phi_2^2(\Delta_2^{-1} + \Delta_3^{-1}) & -n\phi_3\phi_4 / \Delta_4 \\ \text{Symm.} & & n\phi_3^2(\Delta_3^{-1} + \Delta_4^{-1}) \end{bmatrix}$$

Figure 1. The structure of the **Q** with 5 categories.

Building **R:**

The matrix **R** is defined in [24] as an nxn diagonal matrix whose i^{th} diagonal element is $\Sigma_{k-1}^{m} \delta_{ik}^{2} / \Delta_{ik}$. In Building **R** for the first time, all of its diagonal elements are equivalent. Therefore, only one diagonal element of **R** needs to be computed.

Building **ν:**

The vector ν is defined in [19] as an (m-2) x 1 vector whose $(k-1)^{th}$ element is

$$\Sigma_{i \in R_k} (\phi_{ik} / \Delta_{ik}) - \Sigma_{i \in R_{k+1}} (\phi_{ik} / \Delta_{i,k+1}).$$

Here m-2 different values are to be produced. The value of k = 2, corresponds to the first element of ν, therefore individuals of the first category are not used in the computation. The $\Sigma_{i \in R_k}$ is equivalent to the multiplication by the number of individuals in any category, therefore, it was replaced by multiplying by the number of all individuals in the k^{th} category. Figure 2 represents the vector ν if five response categories were of interest.

$$\begin{bmatrix} N_2[(\phi_2 / \Delta_2) - N_3(\phi_2 / \Delta_3)] \\ N_3[(\phi_3 / \Delta_3) - N_4(\phi_3 / \Delta_4)] \\ N_4[(\phi_4 / \Delta_4) - N_5(\phi_4 / \Delta_5)] \end{bmatrix}$$

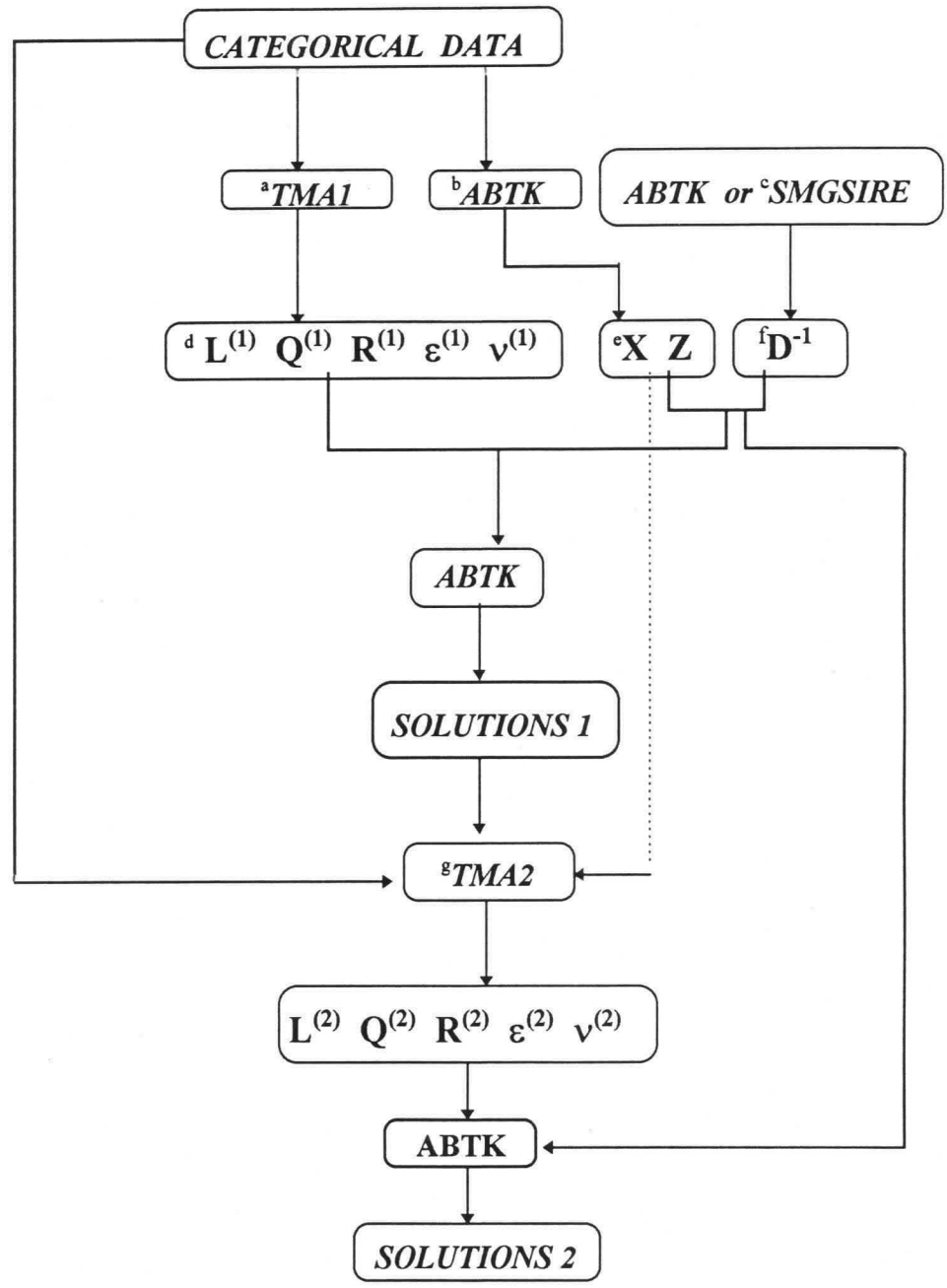
Figure 2. The structure of the ν with 5 categories.

Building ε

The vector ε is defined in [20] as an nx1 vector whose i^{th} element = $\delta_{i,Z_i} / \Delta_{i,Z_i}$, where Z_i is the category into which the i^{th} individual responded. In building ε for the first or later rounds, elements of the vector of categorical observations are to be checked, and based on the response value of the i^{th} individual its corresponding element in ε is computed. The code for the threshold model analysis program is attached in APPENDIX A.

Concatenating the system of the threshold model equations

There are several computer programs available for building the matrices of the mixed part. Animal Breeder's Tool Kit (ABTK) is an appropriate one to use here, Golden et. al.



[a]Computer program to build the matrices of the threshold part for the first time.
[b]Computer program to build the mixed part matrices.
[c]Computer program to build the relationship matrix from sires and maternal grandsires.
[d]Threshold part matrices (or vectors).
[e]Mixed part matrices (both fixed and random effects incidence matrices).
[f]Any type of relationship matrix inverse.
[g] Computer program to build the matrices of the threshold part for second and later time.

Figure 3. Flowchart for using the computer programs developed to analyze categorical data.

(1992). ABTK is a collection of tools used to build and solve the mixed model equations. The program may be used to build the matrices of the mixed part, then horizontally and vertically concatenate all parts together. An iterative tool is available in ABTK to get solutions of a linear system of equations. The equations in the threshold model are nonlinear and need to be built and solved several times. The iterative tool is one way to solve the equations each time. Figure 3 shows the steps involved in building and solving the equations two times.

Building the equations for the second and subsequent rounds

Building the equations for a second and later rounds is basically similar to building them for the first time, however, the computations involved in building the equations for a second time are more complex. Separate routines were written to build the equations for building the equations for any round after the first one. This keeps the routines written for the first round efficient in doing their easier numerical task.

All ABTK tools can be executed interactively as shell commands with command-line arguments, similar to the way the routines developed in the current research can be executed. Therefore, shell scripts can be used to automate the whole process of building and solving the threshold model equations. The code for the second threshold model analysis program is attached in APPENDIX A.

Relationship matrix

Relationship matrix is built for sires or sires and maternal grandsires with a sire model. The complete relationship matrix is needed with an animal model. The current program was designed to handle both. A routine for computing the relationship inverse from sires and maternal grandsires was developed following a rapid algorithm developed by Henderson (1975). The algorithm produces the elements of A^{-1} directly without needing to build the whole relationship matrix and inverting it.

The architecture of the threshold model equations can be built to be either a sire model or an animal model. Of course this also requires building the matrices of the mixed part in a

different manner to satisfy the requirements of each model. See Appendix B for an example of how to get solutions for both sire and animal models.

Validating the computing algorithms

The objective of this section is to use the empirical methods to validate the computing routines. In the simulated data all of the true parameters about the data are known. This knowledge of the true parameters allows us to compare the estimated parameters with their corresponding true parameters.

A categorical data set was simulated for calving ease by Monte Carlo methods. The following model was programmed to generate the data set. Let

$$Y_{ijkl} = \mu + sx_i + pty_j + sr_k + \varepsilon_{ijkl}, \qquad [25]$$

where

Y_{ijkl} is a continuous underlying response variable following mixed linear model and assumed to be normally distributed;

μ is the overall mean;

sx_i is the effect of the i^{th} sex, (i = 1 for a male calf and 2 for a female calf);

pty_j is the effect of the j^{th} parity (j = 1 for the dam's first calf, 2 for her second calf, and 3 for her third and subsequent calf);

sr_k is the effect of the k^{th} sire, (k = 1, . . ., 40); and

ε_{ijkl} is the random residual.

In simulating the data by model [25], the factors μ, sx, and pty were considered fixed factors in the model. The fixed factors have no variance and were assigned constant values. The constant values used were obtained from previous literature (e. g., Harville and Mee, 1984; Djemali,1985; Djemali et. al, 1987).

Sire effect was considered random and normally distributed with mean equal to zero and standard deviation equal to σ_s. Because the heritability model was used, σ_s was computed as 0.5h, where h is the square root of the heritability used, (Ronningen, 1974). In creating the sire distribution, a subroutine for generating uncorrelated random numbers ~N(0, 1) was written and used to generate 40 random numbers. The random numbers were then multiplied

by σ_s to create the sire effect. Each sire effect was repeated 180 times by a finite loop to produce a sire effect common to members of each sire progeny group. This resulted in 7200 sire effects for the whole data set.

In simulating the residual part, a similar approach to that used to create sire effects was followed. The ε_{ijkl} were intended to be a random variable with a normal distribution of mean zero and standard deviation equal to σ_e. The σ_e was computed as $\sqrt{1-0.25h^2}$ according to the heritability model. A total number of 7200 random number was sampled from a standard normal distribution and then multiplied by σ_e. This produces a random error specific to each individual.

Heritability was assumed to be 0.2, then $\sigma_s = 0.5\sqrt{0.2} = 0.224$ and $\sigma_e = \sqrt{1-0.25(0.2)} = 0.975$. The values of 0.224 for the sire standard deviation and .975 for the error standard deviation were used to generate the random factors in the mixed model [25]. Both fixed and random effects were then summed together to compute the underlying continuous response variable, Y_{ijkl}. To compute values of Y_{ijkl} and sire effects(or sire TA's) corresponding to a standardized threshold model, values of the sire effect and Y_{ijkl} were divided by the error standard deviation. The standardized response variable Y_{ijkl} / σ_e were then categorized into five response categories. The distribution of the simulated categorical calving ease data is given in Table 2.

The simulation program, in general, required some input and output. Sire and error standard deviations, constants to be assigned to the fixed effects, and any desired frequency to be used to categorize the continuous underlying response variable are inputs to the program. The program outputs several files contain the simulated sire effects, the true fixed effects and

Table 2. Distribution of simulated calving ease records (N = 7200).

Category	Frequency	Cumulative Percent
1	4320	60.0
2	1080	75.0
3	792	86.0
4	576	94.0
5	432	100.0

boundary points, the sex parity and sire classifications, and the categorical data.

The simulated data were then analyzed by TMA1 to get solutions after building the system of equations only once. The solutions of TMA1, the matrices of the mixed part, and the data were then used as input to the second threshold model analysis program (TMA2) to build the matrices for a second time. This process was repeated by TMA2 six times until the correction became less than 10^{-6}. The repeated process of building and solving the equations can be extended until any desirable convergence criterion is met.

The six sets of estimates and predictions obtained from six iterates were compared to true fixed and random parameters. Sires were ranked based on true and predicted TA's from the six iterates. The rank was also compared among the six sets of sire solutions and true sire TA's.

Application to calving ease

The objective of this section is to present a threshold model procedure for predicting calving ease using TMA. That involves the method and model followed for evaluating sires for the calving ease trait. A big data set with numerous classifications was chosen to test the ability and the efficiency of program.

Data

The data used in this research were taken from the National Association of Animal Breeders (NAAB) calving ease data base. Data were calves born in 1991 to 1995 to active AI sires in December, 1995. Progeny of contemporary herdmate sires to these active AI sires were also included in the data set to give complete herd-year-season calving information on all active AI sires. The data from 1991 to 1995 included 2,371,227 records. A subset of these records was used in the implementation of the computing algorithms. This subset, 215,567 records or one-tenth of the data, was obtained by using every tenth record.

From the subset all sires with less than 5 progeny were deleted, leaving a total number of 206,195 records, 5,593 sires, and 62,758 HYS subclasses. These data had a total number of 68,360 equations.

Models

In the analysis of calving ease data, the following underlying mixed model was assumed

$$Y_{ijkfl} = \mu + SX_i + PTY_j + HYS_k + SR_f + \varepsilon_{ijkfl}, \qquad [26]$$

where

Y is an underlying continuous variable,

μ is the overall mean,

SX_i is the effect of the ith sex (i = 1 for male calves and 2 for female calves),

PTY_j is the effect of the jth parity (j = 1 for dam's first calving, 2 for second calving, and 3 for third and subsequent calving),

HYS_k is the effect of the kth herd-year-season subclass (out of the total of H subclasses, H=62,758),

SR_f is the effect of the f^{th} sire (out of total of S sires, S=5,593), and

ε_{ijkfl} is the residual error.

To overcome the computational problem associated with HYS subclasses containing all individuals with a unique calving score, HYS effect was considered to be a random factor in the model. This solution was suggested by Harville and Mee (1984). Another solution they also suggested was to ignore any HYS subclass having only one category. This solution was not used here because it does not allow the use of all the available data. The model [26] can be written in matrix notation as follows

$$\mathbf{y} = \mathbf{X}\alpha + \mathbf{Z}_1\beta_1 + \mathbf{Z}_2\beta_2 + \varepsilon \qquad [27]$$

where $\mathbf{y}$ is the unobservable underlying continuous response variable; $\mathbf{X}$ is an n x 6 incidence matrix for the fixed effects, sex and parity besides the overall mean; $\mathbf{Z}_1$ is an n x H incidence matrix for the random herd-year-season effects; $\mathbf{Z}_2$ is an n x S incidence matrix for the random sire effects; α is a 6 x 1 vector of fixed unknown parameters; β_1 is an H x 1 vector of unknown random HYS parameters; β_2 is an S x 1 vector of unknown random sire parameters; ε is an nx1 vector of unknown random residual.

The model [27] can be reformulated by subtracting the first boundary point, ξ_1, from both sides of the model equation and dividing both sides by the residual standard deviation, σ_e.

This yields another model, referred to as the standardized threshold model, whose underlying continuous response variables correspond to $(Y_i - \xi_1)\sigma_e^{-1}$, the boundary points correspond to $(\xi_k - \xi_1)\sigma_e^{-1}$, and fixed and random effects correspond to $(\mu - \xi_1 , \alpha_2)\sigma_e^{-1}$ (α_2 is the vector of the unknown fixed parameters pertaining to sex and parity), $\beta_1\sigma_e^{-1}$, $\beta_2\sigma_e^{-1}$, and $\varepsilon\sigma_e^{-1}$, respectively. Reformulating the threshold model into a standardized threshold model decreases the computational tasks required.

To make the assumptions of the standardized threshold model define $\alpha' = [\alpha_1 \; \alpha'_2]$, where α_1 is the unknown fixed parameter pertaining to the overall mean and α'_2 is a vector of the sex and parity unknown fixed parameters; σ_h^2 and σ_s^2 are variances of random herd-year-season and sire, respectively; σ_e^2 is the residual variance; var(**y**) = $\mathbf{Z}_1\mathbf{Z}_1'\sigma_h^2 + \mathbf{Z}_2\mathbf{Z}_2'\sigma_s^2 = \mathbf{V}$, given no covariance among HYS subclasses and unrelated sires; $\mathbf{I}\sigma_h^2$ is the HYS variance covariance matrix; and $\mathbf{I}\sigma_s^2$ is the sire variance covariance matrix.

Given the definitions above, assumptions are

$$E\begin{bmatrix}\mathbf{y}\\ \beta_1\\ \beta_2\\ \varepsilon\end{bmatrix} = \begin{bmatrix}\mathbf{X}\begin{bmatrix}[\alpha_1 - \xi_1]\\ \alpha_2\end{bmatrix}\sigma_e^{-1}\\ \mathbf{0}\\ \mathbf{0}\\ \mathbf{0}\end{bmatrix}, \text{ and } \text{var}\begin{bmatrix}\mathbf{y}\\ \beta_1\\ \beta_2\\ \varepsilon\end{bmatrix} = \begin{bmatrix}\mathbf{v} & \mathbf{Z}_1\sigma_h^2 & \mathbf{Z}_2\sigma_s^2 & \mathbf{I}\sigma_e^2\\ \mathbf{Z}_1'\sigma_h^2 & \mathbf{I}\sigma_h^2 & \mathbf{0} & \mathbf{0}\\ \mathbf{Z}_2'\sigma_s^2 & \mathbf{0} & \mathbf{I}\sigma_s^2 & \mathbf{0}\\ \mathbf{I}\sigma_e^2 & \mathbf{0} & \mathbf{0} & \mathbf{I}\sigma_e^2\end{bmatrix}.$$

Based on the assumptions above, the threshold model equations used are

$$\begin{bmatrix}\mathbf{Q}^k & \mathbf{L}'^k\mathbf{X} & \mathbf{L}'^k\mathbf{Z}_1 & \mathbf{L}'^k\mathbf{Z}_2\\ \mathbf{X}'\mathbf{L}^k & \mathbf{X}'\mathbf{R}^k\mathbf{X} & \mathbf{X}'\mathbf{R}^k\mathbf{Z}_1 & \mathbf{X}'\mathbf{R}^k\mathbf{Z}_2\\ \mathbf{Z}'_1\mathbf{L}^k & \mathbf{Z}'_1\mathbf{R}^k\mathbf{X} & \mathbf{Z}'_1\mathbf{R}^k\mathbf{Z}_1 + \mathbf{D}_1^{-1} & \mathbf{Z}'_1\mathbf{R}^k\mathbf{Z}_2\\ \mathbf{Z}'_2\mathbf{L}^k & \mathbf{Z}'_2\mathbf{R}^k\mathbf{X} & \mathbf{Z}'_2\mathbf{R}^k\mathbf{Z}_1 & \mathbf{Z}'_2\mathbf{R}^k\mathbf{Z}_2 + \mathbf{D}_2^{-1}\end{bmatrix}\begin{bmatrix}\hat{\xi}^{k+1} - \hat{\xi}^k\\ \hat{\alpha}^{k+1} - \hat{\alpha}^k\\ \hat{\beta}_1{}^{k+1} - \hat{\beta}_1{}^k\\ \hat{\beta}_2{}^{k+1} - \hat{\beta}_2{}^k\end{bmatrix} = \begin{bmatrix}\mathbf{t}(\hat{\xi}^k\hat{\alpha}^k\hat{\beta}_1{}^k\hat{\beta}_2{}^k;\mathbf{z})\\ \mathbf{X}'\varepsilon(\hat{\xi}^k\hat{\alpha}^k\hat{\beta}_1{}^k\hat{\beta}_2{}^k;\mathbf{z})\\ \mathbf{Z}'_1\varepsilon(\hat{\xi}^k\hat{\alpha}^k\hat{\beta}_1{}^k\hat{\beta}_2{}^k;\mathbf{z}) - \mathbf{D}_1^{-1}\beta_1\\ \mathbf{Z}'_2\varepsilon(\hat{\xi}^k\hat{\alpha}^k\hat{\beta}_1{}^k\hat{\beta}_2{}^k;\mathbf{z}) - \mathbf{D}_2^{-1}\beta_2\end{bmatrix}, \qquad [28]$$

where, $\mathbf{D}_1^{-1}$ is the inverse HYS variance-covariance matrix and $\mathbf{D}_2^{-1}$ is the inverse sire variance-covariance matrix.

Computational procedures

In solving the non-linear system of threshold model equations above, one matrix $\mathbf{Z} = [\mathbf{Z}_1\ \mathbf{Z}_2]$ was built by horizontally concatenating $\mathbf{Z}_1$ and $\mathbf{Z}_2$ and used in a similar fashion of having only one random factor, however, the structure of the inverse variance-covariance matrix of the random effects was altered to have both HYS and SR inverse variance-covariance matrices by building $\mathbf{D}^{-1} = \begin{bmatrix} \mathbf{D}_1^{-1} & \mathbf{0} \\ \mathbf{0} & \mathbf{D}_2^{-1} \end{bmatrix}$ and then adding $\mathbf{D}^{-1}$ to $\mathbf{Z}'\mathbf{R}^{-1}\mathbf{Z}$.

The equations were built and solved four times. The correction corresponding to the fourth set of solutions was less than 3.10^{-3}. Solutions from the four iterates were compared to each other and to BLUP solutions. Further the total computing time consumed by TMA routines was computed for the given data set.

RESULTS AND DISCUSSION

Validating the computing algorithms

Computing solutions

Solutions for the simulated calving ease data were obtained. Threshold model analysis program (TMA) was used to get successive sets of solutions until convergence was attained. Convergence was attained in the sixth iterate. Table 3 gives the fixed effects solutions for each of the six iterates besides the original fixed effects values used in the simulation. Threshold model equations were solved iteratively for the unknowns several times. The unknowns were used to build the equations for the next time and then the equations were solved iteratively for a second time and so on. In the "inner" iteration no constrains needed to be imposed on the fixed effects to compute solutions.

The original values for the boundary points were determined after some frequency was assumed, Table 2. These values were unknowns and were left to the simulation program to compute based on a given frequency. These values were then output by the simulation program to be compared with the estimated values by TMA. Before the simulation program output the values of the boundary points, the first boundary point was subtracted. This made the first boundary point equal to zero. The estimate for the first boundary point by a standardized threshold model is always zero.

Table 4 gives the original values used for simulating 40 sire effects in the data set. Solutions from the first, fifth, and sixth iterate were also given. Faster convergence was observed for the random effects. There was no need to estimate any kind of variance components for the random solutions because the variance components were known exactly from the simulation. Further, the sires were assumed unrelated, therefore, there was no need to manipulate any kind of relationship matrix.

Quality of a threshold model estimation

Threshold model estimates obtained for three boundary points and two fixed effects are given in Table 5. The estimates obtained for boundary points were very close to the actual values used in the simulation. They differ slightly in the second or the third decimal digit.

Table 3. Fixed effects solutions computed by TMA program in each of 6 iterates compared with true fixed effects in the simulated data.

Parameters	True	Iterate					
		1	2	3	4	5	6
ξ_2	0.481284	0.488727	0.493661	0.493821	0.493827	0.493827	0.493827
ξ_3	0.941494	0.954609	0.969578	0.969934	0.969951	0.969952	0.969952
ξ_4	1.561890	1.488088	1.525608	1.526867	1.526903	1.526905	1.526905
μ	-0.453700	-0.277727	-0.277722	-0.277734	-0.277730	-0.277730	-0.277730
SX_1	0.000000	0.213295	0.216801	0.216879	0.216880	0.216880	0.216880
SX_2	-0.409900	-0.217046	-0.221192	-0.221285	-0.221286	-0.221286	-0.221286
PTY_1	0.993700	0.637432	0.643106	0.643245	0.643250	0.643250	0.643250
PTY_2	0.115900	-0.264125	-0.266345	-0.266353	-0.266353	-0.266353	-0.266353
PTY_3	0.000000	-0.380994	-0.386760	-0.386950	-0.386957	-0.386957	-0.386957

Table 4. Sire solutions from TMA. Iterates, 1, 5, and 6, are compared with true TA's of sires in the simulated data.

Sire	True	Iterate		
		1	5	6
1	-0.440239	-0.295277	-0.300702	-0.300702
2	-0.056049	0.038893	0.039997	0.039997
3	-0.356266	-0.317477	-0.328562	-0.328562
4	-0.747209	-0.717146	-0.762852	-0.762853
5	-0.350707	-0.281985	-0.290035	-0.290035
6	0.379099	0.498319	0.508469	0.508469
7	0.170602	-0.061142	-0.059256	-0.059256
8	-0.168094	-0.173621	-0.178403	-0.178403
9	-0.133690	-0.160827	-0.162479	-0.162479
10	-0.140379	-0.053986	-0.058027	-0.058027
11	0.163185	0.184491	0.191739	0.191739
12	-0.244709	-0.138725	-0.141105	-0.141105
13	-0.118642	-0.058834	-0.059386	-0.059386
14	-0.235895	-0.227982	-0.237458	-0.237458
15	-0.251116	-0.236301	-0.241155	-0.241155
16	0.056803	0.185779	0.187150	0.187150
17	0.477463	0.447878	0.459973	0.459973
18	-0.124796	-0.044975	-0.043659	-0.043659
19	-0.096308	-0.112885	-0.113208	-0.113208
20	-0.202378	-0.066295	-0.067418	-0.067418
21	-0.061961	-0.037681	-0.036529	-0.036529
22	0.154167	0.246992	0.253418	0.253418
23	0.344806	0.320885	0.330492	0.330492
24	0.045928	0.019550	0.021258	0.021258
25	-0.022465	0.073168	0.077617	0.077617
26	0.401948	0.453205	0.466455	0.466455
27	0.089219	0.090075	0.095131	0.095131
28	0.014432	-0.068544	-0.068791	-0.068791
29	-0.208210	-0.105802	-0.105457	-0.105457
30	0.349720	0.458203	0.467271	0.467271
31	0.000273	-0.065163	-0.066495	-0.066494
32	0.220934	0.282516	0.289665	0.289665
33	-0.149818	-0.123437	-0.123697	-0.123697
34	-0.081273	-0.056094	-0.057784	-0.057784
35	-0.072009	-0.052030	-0.049661	-0.049661
36	0.165030	0.132089	0.135042	0.135042
37	-0.327447	-0.178663	-0.181606	-0.181606
38	-0.283565	-0.206230	-0.208809	-0.208809
39	0.242878	0.291513	0.297932	0.297932
40	0.072596	0.117490	0.120925	0.120925

Little change was found in solutions obtained from the convergent (the sixth) iterate compared with solutions obtained from the first iterate.

Differences among estimates obtained for fixed effects were almost equivalent to those used to simulate the data. Slight change was noticed between the first and the sixth iterate. This change did not seem to improve the quality of the solutions. By looking at the differences given in Table 5, one can notice that differences of iterate number 6 are not closer to the true differences than differences of iterate 1.

In the matrix of Figure 4, correlation coefficients among true and estimated fixed effects and boundary points are given. The correlation coefficients computed among solutions

Table 5. Comparison of differences among true and estimated fixed effects by TMA. Results of the 1st and 6th iterates are given.

Difference	True	Iterate	
		1	6
$SX_1 - SX_2$	0.409900	0.430341	0.438166
$PTY_1 - PTY_2$	0.877800	0.901557	0.909603
$PTY_1 - PTY_3$	0.993700	1.018426	1.030207
$PTY_2 - PTY_3$	0.115900	0.116869	0.120604

	[a] true	[b] it.1	it.2	it.3	it.4	it.5	it.6
true	1.0000	0.9319	0.9316	0.9316	0.9316	0.9316	0.9316
it.1		1.0000	1.0000	1.0000	1.0000	1.0000	1.0000
it.2			1.0000	1.0000	1.0000	1.0000	1.0000
it.3				1.0000	1.0000	1.0000	1.0000
it.4					1.0000	1.0000	1.0000
it.5						1.0000	1.0000
it.6							1.0000

[a] Simulated fixed effects.
[b] Iterate number 1.

Figure 4. Correlation coefficients among true and six sets of estimated fixed effects from six iterates of TMA.

obtained from six boundary points are all 1.0000's. Fairly high correlation was also noticed between the true and estimated values.

Quality of a threshold model prediction

Sire predictions were computed by TMA program. Solutions obtained from iterates, 1, 5, and 6 for the 40 sires of the simulated data are given in Table 4. True values used to simulate sire TA's are also given. Correlation coefficient of 0.95 was found between true TA's and predicted values obtained from the first iterate. That correlation changed slightly after the first round. It became 0.96 in the sixth iterate (Figure 5). Correlation coefficients among the sets of solutions obtained from six iterates were almost 1.0000's.

By having a closer look at the change of the predictions from one round to another, one can find that this change somehow improved these predictions by moving them in the correct direction towards the true values.

Table 6 presents the sire rank based on each of the true and predicted values from iterate 1, and 6. By comparing the rank based on the true values and on predicted values it was found that, at least the upper and lower 12.5 % of the sires ranked the same in each of the

	[a] true	[b] it.1	it.2	it.3	it.4	it.5	it.6
true	1.0000	0.9541	0.9588	0.9589	0.9589	0.9589	0.9589
it.1		1.0000	0.9999	0.9999	0.9998	0.9998	0.9998
it.2			1.0000	1.0000	1.0000	1.0000	1.0000
it.3				1.0000	1.0000	1.0000	1.0000
it.4					1.0000	1.0000	1.0000
it.5						1.0000	1.0000
it.6							1.0000

[a] Simulated sire effects.
[b] Iterate number.

Figure 5. correlation coefficients among true and six sets of predicted sire random effects from six iterates of TMA.

Table 6. Comparison of sire rank by simulated and predicted TA's. Ranks based upon predictions from iterates, 1, and 6 are presented.

Position	Rank by		
	True	1st iterate	6th iterate
1	4	4	4
2	1	3	3
3	3	1	1
4	5	5	5
5	37	15	15
6	38	14	14
7	15	38	38
8	12	37	37
9	14	8	8
10	29	9	9
11	20	12	12
12	8	33	33
13	33	19	19
14	10	29	29
15	9	28	28
16	18	20	20
17	13	31	31
18	19	7	13
19	34	13	7
20	35	34	10
21	21	10	34
22	2	35	35
23	25	18	18
24	31	21	21
25	28	24	24
26	24	2	2
27	16	25	25
28	40	27	27
29	27	40	40
30	22	36	36
31	11	11	16
32	36	16	11
33	7	22	22
34	32	32	32
35	39	39	39
36	23	23	23
37	30	17	17
38	6	26	26
39	26	30	30
40	17	6	6

* Shaded areas highlight sires of similar rank.

three lists. Some sires in the middle part changed rank, but no drastic change in rank was found. As indicated in the simulation study presented in the next chapter, identical rank by true and predicted TA's is not expected. Similar rank is unlikely unless the heritability of the trait being simulated is high, number of daughters per sire is large, the incidence of the trait is moderate or good (not extreme), and multiple categorization is used.

Solutions from both iterate 1, and 6 ranked sires similarly. The largest change switched two sires only one position. Sires 13 and 7, 10 and 34, and 16 and 11 switched one position, the rest of the 40 sires had exactly the same rank. This indicates that even in computing random solutions "outer" convergence did not add much more information beyond the first iterate.

History of convergence

The unknowns in the threshold model equations are differences between two successive sets of solutions and not the solutions themselves, i.e, $\xi^{(k+1)} - \xi^{(k)}$, $\alpha^{(k+1)} - \alpha^{(k)}$, and, $\beta^{(k+1)} - \beta^{(k)}$. The iterative algorithm [21] was solved several times for these differences, called corrections, and then they were added to the former set of solutions before being used in building the equations for another time. This process of building the equations several times was stopped when the corrections became less than 10^{-6} in all of the unknowns. Correction reached this value after building the equations six times. TMA1 program was used for building the equations for the first time, then TMA2 was used to build them for the second and subsequent times. The values of the corrections obtained after solving the equations six times were detected because convergence is decided based on the magnitude of these values. Corrections are given in Table 7 for estimates of the boundary points and fixed effects and in Table 8 for random effects. The corrections were computed in each of the six iterates by solving the system of threshold model equations iteratively after the equations were built from solutions from the previous iterate.

Corrections obtained from the first iterate were as small as 10^{-1} to 10^{-3}. Corrections for the boundary points, especially the last one, converged rather slowly, notice that this boundary

point identifies the least frequent category in the data set. In contrast, the fastest converging solution observed was for the mean.

Calving ease data

Total number of 206,195 records were available for analysis by TMA program. The objective was to expose the program to relatively large data set with a big number of classifications. A total of 5,593 sires and 62,758 HYS subclasses resulted in 68,351 equations, fixed effects equations are not included in this number. The large number of HYS subclasses increases the possibility of having some subclasses with individuals of only one category for each subclass. This causes a computational problem as mentioned earlier. Considering HYS as a random effect in the threshold model is one solution to the problem. Table 9 and 10 summarize important categorical properties of the data available.

Threshold model solutions

The steps followed to get solutions are given in the shell script presented in APPENDIX B. TMA program was used to compute threshold model solutions from four iterates. The fixed-parameter estimates are shown in Table 11. The iteration was stopped when the correction became less than 4.10^{-3} (Table 12).

The variance ratios for the random effects were taken from previous literature. Variances of HYS, sire, and residual effects were available for similar calving ease data(e.g., Harville and Mee, 1984; Djemali, 1985; Clutter et al., 1989). It was not readily possible to compute estimates of variance components for 68,351 random equations by any available threshold model-estimation computer program because of array-size limitations of such programs (e.g., [1]CMMAT program). The values used for the variance of sire and HYS effects were .0385 and .0962, respectively, corresponding to heritability equal to 0.148 in the underlying scale. Estimating variance components was not our objective in this research.

---

[1] FORTRAN program written by I. Misztal

Table 7. History of convergence over six iterates for fixed effect solutions estimated by TMA.

Correction	k[a]					
	0	1	2	3	4	5
$\xi_2^{(k+1)} - \xi_2^{(k)}$	0.067584	0.004933	0.000160	0.000006	0.000000	0.000000
$\xi_3^{(k+1)} - \xi_3^{(k)}$	0.127637	0.014969	0.000356	0.000017	0.000000	0.000000
$\xi_4^{(k+1)} - \xi_4^{(k)}$	0.186665	0.037520	0.001259	0.000036	0.000002	0.000000
$\mu^{(k+1)} - \mu^{(k)}$	-0.024671	0.000005	-0.000012	0.000004	0.000000	0.000000
$SX_1^{(k+1)} - SX_1^{(k)}$	0.024410	0.003505	0.000078	0.000001	0.000000	0.000000
$SX_2^{(k+1)} - SX_2^{(k)}$	-0.028138	-0.004146	-0.000093	-0.000001	0.000000	0.000000
$PTY_1^{(k+1)} - PTY_1^{(k)}$	0.019065	0.005674	0.000139	0.000005	0.000000	0.000000
$PTY_2^{(k+1)} - PTY_2^{(k)}$	0.001059	-0.002220	-0.000009	0.000000	0.000000	0.000000
$PTY_3^{(k+1)} - PTY_3^{(k)}$	-0.026960	-0.005766	-0.000190	-0.000007	0.000000	0.000000

[a] The k^{th} iterate

Table 8. History of convergence over six iterates for random effect solutions computed by TMA.

Sire	$\hat{\beta}^{(k+1)} - \hat{\beta}^{(k)}$					
	k = 0	k = 1	k = 2	k = 3	k = 4	k = 5
1	-0.044872	-0.005226	-0.000192	-0.000008	0.000000	0.000000
2	0.016512	0.001118	-0.000014	0.000000	0.000000	0.000000
3	-0.057033	-0.010262	-0.000796	-0.000026	-0.000001	0.000000
4	-0.174425	-0.042129	-0.003390	-0.000179	-0.000008	0.000000
5	-0.044007	-0.007281	-0.000740	-0.000027	-0.000001	0.000000
6	0.021688	0.009264	0.000862	0.000024	0.000001	0.000000
7	-0.000653	0.001248	0.000605	0.000031	0.000001	0.000000
8	-0.007399	-0.004270	-0.000497	-0.000014	-0.000001	0.000000
9	0.004826	-0.001345	-0.000295	-0.000011	0.000000	0.000000
10	0.006770	-0.003157	-0.000871	-0.000012	-0.000001	0.000000
11	0.024032	0.006750	0.000491	0.000008	0.000000	0.000000
12	0.004644	-0.002113	-0.000258	-0.000008	0.000000	0.000000
13	0.002203	-0.000479	-0.000072	-0.000001	0.000000	0.000000
14	-0.032274	-0.008464	-0.000990	-0.000021	-0.000001	0.000000
15	-0.020350	-0.004298	-0.000540	-0.000016	-0.000001	0.000000
16	0.020791	0.001555	-0.000181	-0.000003	0.000000	0.000000
17	0.026233	0.010942	0.001117	0.000035	0.000001	0.000000
18	0.004787	0.001641	-0.000314	-0.000012	0.000000	0.000000
19	-0.005474	-0.000594	0.000257	0.000014	0.000001	0.000000
20	0.000030	-0.001232	0.000101	0.000008	0.000000	0.000000
21	0.017368	0.001423	-0.000261	-0.000009	0.000000	0.000000
22	0.025302	0.006210	0.000214	0.000002	0.000000	0.000000
23	0.024570	0.008861	0.000733	0.000013	0.000001	0.000000
24	0.000364	0.001846	-0.000132	-0.000006	0.000000	0.000000
25	0.014958	0.003705	0.000711	0.000033	0.000001	0.000000
26	0.028335	0.011585	0.001602	0.000061	0.000002	0.000000
27	0.029437	0.004281	0.000748	0.000025	0.000001	0.000000
28	0.010240	-0.000770	0.000488	0.000034	0.000001	0.000000
29	0.002405	0.000074	0.000259	0.000011	0.000000	0.000000
30	0.029730	0.007976	0.001044	0.000045	0.000001	0.000000
31	0.000464	-0.001208	-0.000120	-0.000005	0.000000	0.000000
32	0.021492	0.006617	0.000516	0.000015	0.000001	0.000000
33	-0.006587	-0.000428	0.000161	0.000007	0.000000	0.000000
34	0.010748	-0.001299	-0.000381	-0.000010	0.000000	0.000000
35	0.016404	0.002279	0.000091	-0.000001	0.000000	0.000000
36	0.019819	0.002635	0.000304	0.000014	0.000000	0.000000
37	-0.013654	-0.003066	0.000113	0.000010	0.000000	0.000000
38	-0.022215	-0.002173	-0.000390	-0.000016	0.000000	0.000000
39	0.019526	0.006334	0.000085	0.000000	0.000000	0.000000
40	0.025053	0.003510	-0.000072	-0.000003	0.000000	0.000000

Table 9. Distribution of five categories for calving ease with respect to sex of calf, parity of dam, and over all effects.

Category	Sex %		Parity %			Total %
	Male	Female	1	2	≥ 3	
1	38.2	40.2	17.0	23.4	38.0	78.4
2	5.2	4.2	3.6	2.3	3.4	9.3
3	4.8	3.1	3.7	1.7	2.6	8.0
4	1.9	.9	1.5	.6	.8	2.9
5	1.1	.4	.8	.3	.4	1.5

Table 10. Frequency distribution of five response categories within sex of calf and parity of dam.

		Male		Female	
Parity	Category	Number	Percent	Number	Percent
1	1	15396	14.57	19604	19.50
	2	3927	3.72	3565	3.55
	3	4581	4.33	3034	3.02
	4	2034	1.92	1022	1.02
	5	1157	1.09	427	0.42
	Subtotal	27095	25.63	27652	27.51
2	1	23917	22.63	24335	24.21
	2	2645	2.50	2028	2.02
	3	2158	2.04	1279	1.27
	4	830	0.79	368	0.37
	5	462	0.44	152	0.15
	Subtotal	30012	28.40	28162	28.02
3	1	39485	37.36	38941	38.74
	2	4069	3.85	2989	2.97
	3	3245	3.07	2019	2.01
	4	1131	1.07	524	0.52
	5	639	0.60	232	0.23
	Subtotal	48569	45.95	44705	44.47
	Total	105676	100.00	100519	100.00

Efficiency of the programs

Programs TMA1 and TMA2 were compiled on DECstation 5000/25 (OS Version: ULTRIX V4.3A). The time consumed by TMA1 to build the threshold matrices for the whole data set (206,195 records in 68,360 equations) in the first round was 1-2 minutes. The time consumed by TMA2 to build the threshold matrices for second or later rounds was 10-14 minutes. The programs are considered efficient in terms of time. Building the threshold model equations involves a tremendous number of numerical tasks (e.g., a large number of normal densities and probability integrals are required).

The programs are also efficient in terms of disk space consumption. The storage mode of the output matrices is sparse or diagonal depending on the structure of the matrix. For example the matrix **R** is always produced in a diagonal mode, i.e., only the diagonal elements of **R** are stored.

Threshold model solutions compared with BLUP solutions

As explained earlier, BLUP is not appropriate for application to sire evaluation for threshold traits. BLUP solutions were computed with a sire model for the same calving ease data to be compared with threshold model solutions. HYS was considered a random effect in the mixed model used, following a similar procedure to the threshold model analysis. Values for the variance components were obtained from a similar previous study on calving ease (Djemali et al., 1987). The values used were .0668, .0084, and .5452 for herd-year-season, sire, and residual effects, respectively.

The correlation between BLUP solutions and threshold model solutions for sires from four iterates was computed. The correlation coefficients were about 0.95. The correlation coefficients among the four sets of solutions obtained from four iterates of TMA were also computed, they varied from 0.9926 to 0.9999. Table 13 summarizes simple statistics for BLUP solutions and TMA solutions. Only sire solutions are compared.

Table 11. Fixed parameter estimates.

[a]Parameter	Iterate				BLUP
Estimates	1	2	3	4	
ξ_2	0.375336	0.431794	0.437991	0.440084	
ξ_3	0.924587	1.044019	1.069834	1.073803	
ξ_4	1.386755	1.530762	1.591123	1.592636	
μ	-0.796699	-0.867190	-0.869452	-0.869796	1.394560
SX_{12}	-0.153350	-0.168178	-0.172996	-0.173764	-0.100158
SX_{21}	0.146064	0.157733	0.161401	0.161982	0.095270
PTY_1	0.500924	0.487920	0.492806	0.493292	0.309670
PTY_{23}	-0.195426	-0.191865	-0.195080	-0.195417	-0.120610
PTY_{32}	-0.158592	-0.146439	-0.148219	-0.148371	-0.098047

[a] Estimated parameters corresponding to the standardized threshold model.

Table 12. History of convergence over four iterates for fixed effects' solutions.

Correction	k[a]			
	0	1	2	3
$\xi_2^{(k+1)} - \xi_2^{(k)}$	-0.000306	0.056458	0.006197	0.002094
$\xi_3^{(k+1)} - \xi_3^{(k)}$	-0.000590	0.119432	0.025815	0.003969
$\xi_4^{(k+1)} - \xi_4^{(k)}$	-0.000655	0.144007	0.060361	0.001513
$\mu^{(k+1)} - \mu^{(k)}$	-0.010505	-0.070491	-0.002262	-0.000345
$SX_{12}^{(k+1)} - SX_{12}^{(k)}$	-0.153350	-0.014828	-0.004818	-0.000768
$SX_{21}^{(k+1)} - SX_{21}^{(k)}$	0.146064	0.011669	0.003668	0.000581
$PTY_1^{(k+1)} - PTY_1^{(k)}$	0.500924	-0.013004	0.004887	0.000486
$PTY_{23}^{(k+1)} - PTY_{23}^{(k)}$	-0.195426	0.003561	-0.003215	-0.000337
$PTY_{32}^{(k+1)} - PTY_{32}^{(k)}$	-0.158592	0.012153	-0.001780	-0.000152

[a] The k^{th} iterate.

Table 13. Summary of sire solutions from four iterates of Threshold Model Analysis and BLUP.

Solution	N	Mean	Std. Dev.	Min.	Max.
it.1	5593	3.45 E-7	0.078	-0.302	0.575
it.2	5593	-1.25 E-8	0.080	-0.318	0.546
it.3	5593	3.40 E-9	0.083	-0.327	0.554
it.4	5593	-2.32 E-9	0.084	-0.330	0.555
BLUP	5593	6.26 E-9	0.039	-0.200	0.354

VALIDITY AND PROPERTIES OF NONLINEAR METHODS OF SIRE EVALUATION FOR THRESHOLD TRAITS

A paper to be submitted to the Journal of Dairy Science

Gamal A. Abdel-Azim, P. J. Berger

INTRODUCTION

Animal breeding data are measured on one of two scales, continuous or discrete. Many traits of importance in animal breeding such as calving ease, disease resistance, livability, sex of calf, etc. are categorical in expression, hence, they are measured on discrete scale. Sire evaluation for both types of traits is achieved through different methodology. If the response variable is continuous, normally distributed, and can be represented by a fixed, mixed or random linear statistical model, best linear unbiased prediction (BLUP) is the best method of evaluation (5). Categorical variables, on the other hand, violate many of the required assumptions for mixed models, therefor, BLUP or any other linear method is not appropriate for use (1, 9).

Based on the threshold concept, many non-linear methods have been described for sire evaluation for categorical traits (1, 2, 3, 4). The threshold model methods are based on the assumption of an underlying unobservable continuous response variable that follows a mixed linear model (2, 4).

Although the application of BLUP to categorical data is not justified as most of the assumptions are violated, several studies have compared solutions computed by a linear mixed model with solutions computed by a nonlinear threshold mixed model. In some cases both BLUP and threshold model (TM) gave similar response to selection, but in other cases when incidence of categories became more extreme, when heritability increased, and when the number of response categories decreased, TM gave higher response to selection than BLUP (6, 7). The similarity observed between solutions computed by both methods in some cases could be due to the variance components used. The variance ratio used in BLUP equations was computed based on the threshold concept. BLUP, however, does not assume any underlying scale, therefore the variance ratio is to be computed based on the outward scale for

BLUP as in any continuous data. On the other hand, estimates of variance components by a threshold model are based on the threshold concept and they differ from estimates obtained by a linear mixed model. The difference in variance component estimates could be a significant factor in making threshold model solutions vary from linear model solutions.

Comparing TM and BLUP solutions for a set of data is not sufficient to study the properties and the validity of TM, especially because the comparison needs further implementations. To validate TM, its accuracy of prediction has to be studied by comparing true sire transmitting abilities (TA's) with estimated TA's by TM. True sire TA's are not observable except in computer-simulated data.

The objectives of this research were to study the validity of TM procedure and to investigate the effect of each of heritability, number of daughters per sire, number of categories, and incidence of categories on TM accuracy of prediction. Data were simulated using Monte Carlo techniques (8).

MATERIALS AND METHODS

Programs

1. Threshold model analysis program

A computer program for computing threshold model estimates and predictions was written and applied to the current study. The program can be used to build the threshold model equations, [2], proposed by Harville and Mee (1984). In developing the threshold model equations an underlying scale was to be assumed. The underlying scale follows the linear mixed model, [1].

$$\mathbf{y} = \mathbf{X}\alpha + \mathbf{Z}\beta + \mathbf{e}, \qquad [1]$$

where, $\mathbf{X}$ and $\mathbf{Z}$ are N x p and N x q incidence matrices for fixed and random effects, respectively; α and β are p x 1 and q x 1 vectors of unknown fixed and random parameters, respectively; $\mathbf{e}$ is an N x 1 vector of random residuals; $\mathbf{y}$ is an N x 1 vector of continuous response values, Y_i, i = 1, . . ., N. It is assumed that Y_i itself is not observable, rather the category to which the i^{th} individual belongs, say Z_i, is observable and has the following relationship with Y_i

$\xi_{k-1} < Y_i \leq \xi_k$, where,

k is the category into which the i^{th} individual falls and ξ_k is the k^{th} boundary point.

The threshold model equations are written as follows

$$\begin{bmatrix} \mathbf{Q}^k & \mathbf{L}^k\mathbf{X} & \mathbf{L}^k\mathbf{Z} \\ \mathbf{X}'\mathbf{L}^k & \mathbf{X}'\mathbf{R}^k\mathbf{X} & \mathbf{X}'\mathbf{R}^k\mathbf{Z} \\ \mathbf{Z}'\mathbf{L}^k & \mathbf{Z}'\mathbf{R}^k\mathbf{X} & \mathbf{Z}'\mathbf{R}^k\mathbf{Z}+\mathbf{D}^{-1} \end{bmatrix} \begin{bmatrix} \hat{\xi}^{k+1}-\hat{\xi}^k \\ \hat{\alpha}^{k+1}-\hat{\alpha}^k \\ \hat{\beta}^{k+1}-\hat{\beta}^k \end{bmatrix} = \begin{bmatrix} \mathbf{t}(\hat{\xi}^k\hat{\alpha}^k\hat{\beta}^k;\mathbf{z}) \\ \mathbf{X}'\varepsilon(\hat{\xi}^k\hat{\alpha}^k\hat{\beta}^k;\mathbf{z}) \\ \mathbf{Z}'\varepsilon(\hat{\xi}^k\hat{\alpha}^k\hat{\beta}^k;\mathbf{z})-\mathbf{D}^{-1}\beta \end{bmatrix} \qquad [2]$$

Explicit expressions for the elements of **Q**, **L**, **R**, **t**, and ε are given in (4). $\hat{\xi}^k$, $\hat{\alpha}^K$, and $\hat{\beta}^k$ are vectors of estimates obtained after building the equations k times for (L-2) boundary points (L is the number of response categories), p fixed parameters, and q random parameters, respectively. The equations in [2] are solved for the corrections $\hat{\xi}^{k+1}$-$\hat{\xi}^k$, $\hat{\alpha}^{k+1}$-$\hat{\alpha}^k$, and $\hat{\beta}^{k+1}$-$\hat{\beta}^k$ then solutions are added to the previous iterate. This process is repeated until the corrections get to zero. Initial guesses for $\hat{\xi}^0$, $\hat{\alpha}^0$, and $\hat{\beta}^0$ are computed to build the system of equations for the first time. In the current study the same coefficient matrix was used each time to reduce the total amount of computations (4).

Estimates for the boundary points, and fixed effects can be computed by solving the threshold model equations. Further, predictions for the random effects in the model can be computed as in the conventional mixed model procedure except that none of the required assumptions are violated.

2. Simulation program

A heritability model was programmed to generate the required data for this study. The model can be expressed as

$$y_{ijkl} = \mu + SX_i + PTY_j + SR_k + e_{ijkl}, \qquad [3]$$

where, y_{ijkl} is a continuous response variable; μ is the overall mean; SX_i is a fixed effect, represents the sex factor, i = 1, 2; PTY_j is a fixed effect, represents the parity factor, i = 1, 2, 3; SR_k is a random effect, represents the sire factor, k = 1, . . ., 30; and e_{ijkl}, is a random residual.

Sex and parity factors were considered fixed and assigned constants values. Sire effect was a random factor in the model having a variance. Sire effects were generated as $(\sigma_s R_1)_k$, where σ_s is the sire standard deviation multiplied by R_1 to generate an effect common to members of the k^{th} group. Let h^2 represent heritability of the trait being simulated, then σ_s can be computed as $0.5\sqrt{h^2}$, so that it varies according to a given heritability. The random residual is computed as $(\sigma_e R_2)_{ijkl}$, where σ_e the standard deviation for the error, it was computed as $\sqrt{1-0.25h^2}$, then multiplied by R_2 to create a random error associated with each individual. R_1 and R_2 are random numbers follow a standard normal distribution.

The underlying continuous variable y_{ijkl} was then produced by adding all the generated effects in [3]. Both sides of [3] were divided by the standard deviation of the error to make an error variance equal to 1.0. The underlying variable was categorized after being built according to some frequency. The frequency used to categorize the underlying variable was determined based on the required incidence for the trait being simulated. The boundary points were determined after applying a frequency to the continuous variable then the first boundary point was subtracted from the other boundary points to make the first boundary point equal to 0.0. This process of dividing by the error standard deviation and subtracting the first boundary point produced a standardized response variable with variance equal to 1 and first boundary point equal to 0.

Data and experimental design

Two different settings were designed. Setting I. The emphasis here was to investigate the effect of heritability, categorization type (binary, multiple), and progeny group size on accuracy of genetic prediction. Five Monte Carlo replicates for each of twelve combinations of three levels of σ_s (.1, .22, and .35), two categorization types, and two progeny group sizes were simulated. The two types of categorizations used were: 1) Binary: the continuous variable **y** was dichotomized at a boundary point which allowed an incidence of 90% and 10% for two categories; and 2) Multiple: **y** was polychotomized into five categories of incidence 60%, 15%, 11%, 8%, and 6%. The two progeny group sizes were: 1) Large number

of progeny per sire, where 360 daughters were assigned to each of 30 sires; and 2) Small and variable number, which varied from 12 to 288 daughters per sire.

Setting II. The emphasis was to study the effect of different percentage incidences of the categories on the accuracy of prediction. Three data sets in each of five replicates were generated. Binary categorized variable was simulated. The incidence of category 1 in data set 1, 2, and 3 was 90%, 95%, and 99%, respectively. Large half-sib families of 360 daughters per sire were used.

Models

The simulation model, [3], can be represented in matrix notation as follows

$$\mathbf{y} = \mathbf{X}\alpha + \mathbf{Z}\beta + \mathbf{e},$$

where, **X** and **Z** are known incidence matrices of dimensions n x 6 and n x 30 respectively, α is 6 x 1 vector of unknown fixed effects, and β is 30 x 1 vector of unknown random effects or actual sire transmitting abilities. Assumptions were

$$E\begin{bmatrix}\mathbf{y}\\ \beta\\ \mathbf{e}\end{bmatrix} = \begin{bmatrix}\mathbf{X}\alpha\\ \mathbf{0}\\ \mathbf{0}\end{bmatrix}, \text{ and } \text{var}\begin{bmatrix}\mathbf{y}\\ \beta\\ \mathbf{e}\end{bmatrix} = \begin{bmatrix}\mathbf{ZZ}'\sigma_s^2 + \mathbf{I}\sigma_e^2 & \mathbf{Z}\sigma_s^2 & \mathbf{I}\sigma_e^2\\ & \mathbf{I}\sigma_s^2 & 0\\ \text{Symm.} & & \mathbf{I}\sigma_e^2\end{bmatrix}$$

After standardizing the continuous variable **y** by subtracting ξ_1 and dividing by σ_e as described earlier, the reformulated model became

$$\mathbf{w} = [\mathbf{1}\ \mathbf{X}]\begin{bmatrix}-\xi_1\\ \alpha\end{bmatrix}\sigma_e^{-1} + \mathbf{Z}[\sigma_e^{-1}\beta] + \sigma_e^{-1}\mathbf{e}, \qquad [4]$$

where **w** is the new continuous response variable corresponding to $(Y_i - \xi_1) / \sigma_e$, $i = 1, \ldots, n$, and whose new boundary points correspond to $(\xi_k - \xi_1) / \sigma_e$, $k = 1, 2, \ldots, L$, where, L is the number of categories in the categorical response variable.

New assumptions for the standardized threshold model are

$$E\begin{bmatrix}\mathbf{y}\\ \boldsymbol{\beta}\\ \mathbf{e}\end{bmatrix} = \begin{bmatrix}\mathbf{X}\begin{bmatrix}[\alpha_1 - \xi_1]\\ \alpha_2\end{bmatrix}\sigma_e^{-1}\\ \mathbf{0}\\ \mathbf{0}\end{bmatrix}, \text{ and } \operatorname{var}\begin{bmatrix}\mathbf{y}\\ \boldsymbol{\beta}\\ \mathbf{e}\end{bmatrix} = \begin{bmatrix}\mathbf{ZZ}'\theta + \mathbf{I} & \mathbf{Z}\theta & \mathbf{I}\\ & \mathbf{I}\theta & \mathbf{0}\\ & & \mathbf{I}\end{bmatrix},$$

where $\theta = \frac{\sigma_s^2}{\sigma_e^2}$, and $\alpha' = [\alpha_1 \ \alpha_2']$.

Standardization of the model as mentioned above alleviates the computations, because the parts of **Q**, **L**, **R**, **t**, and ε pertaining to ξ_1 need not be computed.

Methods of validation

The threshold model equations, [1], were used to obtain solutions for all simulated data sets in setting I and II. Sire solutions were taken as the estimated sire transmitting abilities. To compare data sets within setting I and II, the correlation between actual and estimated transmitting abilities was computed. The correlation coefficient was computed for each one of the five replicates generated for each data set, then the correlation coefficients of these five replicates were averaged. Two types of correlation coefficients were computed, Spearman and Pearson. Spearman correlation indicates the correctness of sire ranking, and Pearson correlation indicates the accuracy of prediction of sire transmitting abilities. As an additional comparison mean, response to selection was computed for data sets using the formula

$$R = \sigma_{A\hat{A}}.i.\sigma_A, \qquad [5]$$

where, R denotes response to selection, $\sigma_{A\hat{A}}$ denotes average correlation coefficient computed for each data set, i is selection intensity, and σ_A is average additive genetic standard deviation computed from five replicates of each data set.

RESULTS

Table 1 gives the average values of correlation coefficients of five replicates for each of the twelve combinations of three levels of heritability, two progeny group sizes, and two

categorization types in setting I. It is obvious that the correlations between true and predicted transmitting abilities increased as heritability, number of daughters per sire, and number of categories increased. It is also clear that the dichotomized variable gave lower values of rank correlation in both large and small progeny groups. Table 1 also shows the values of response to selection for the twelve combinations of setting I. The data set of large families, polychotomous response, and high h^2 (σ_s = .35) gave the highest response. The data set of small families, dichotomous response, and low h^2 (σ_s = .10) gave the lowest response to selection.

It appears that all three factors, h^2, number of categories, and family size behaved consistently in all combinations of setting I without interaction. For example, multiple categorization type always gave higher correlations than the binary categorized variable with all three sire variances (Table 1).

Table 1. Comparison of acuracy of prediction and response to selection among twelve combinations of three levels of heritability, two progeny group sizes, and two categorization types.

σ_s	Correlation[1]		ATA[2]		ETA[3]		R[4]
	Spearman	Pearson	μ	SD	μ	SD	
Large families and dichotomous response							
.10	.780	.840	-.006	.115	1.4E-7	.077	.135
.22	.856	.891	-.020	.168	2.3E-7	.152	.209
.35	.959	.959	-.023	.430	2.3E-6	.416	.578
Large families and polychotomous response							
.10	.865	.916	-.018	.124	8.2E-8	.091	.159
.22	.941	.980	-.040	.279	7.2E-6	.246	.383
.35	.981	.985	-.066	.465	3.1E-6	.406	.641
Small families and dichotomous response							
.10	.536	.585	.008	.109	-1.2E-7	.046	.089
.22	.742	.750	.018	.244	-7.5E-7	.142	.256
.35	.801	.813	.029	.407	1.4E-6	.277	.463
Small families and polychotomous response							
.10	.591	.602	.005	.102	6.9E-8	.056	.085
.22	.796	.812	.011	.228	7.2E-7	.166	.259
.35	.870	.893	.017	.385	-2.0E-6	.288	.481

[1] Correlation between predicted and true sire TA's in the simulated data.
[2] Means and standard deviations of actual transmitting abilities (average of five replicates).
[3] Means and standard deviations of estimated transmitting abilities (average of five replicates).
[4] Response to selection of the highest 20%; computed using equation [5], (i = 1.4).

Table 2 gives the average value of correlation coefficients of the five replicates for each of the three data sets of setting II. Correlation measurements decreased as incidence became more extreme.

One distinctive feature of these results is the similarity among replicates. Conclusions stated about the trend of change in the correlation coefficients on the average level were also true on the individual replicates level. In other words, all replicates gave exactly consistent trends of change in accuracy of prediction.

Table 2. Comparison between true and predicted transmitting ability at different levels of incidence.

Incidence	Correlation[1]		ATA[2]		ETA[3]		R^4
	Spearman	Pearson	μ	SD	μ	SD	
90%	.951	.935	.045	.211	-9.2E-7	.161	.276
95%	.781	.822	.045	.211	-2.8E-7	.156	.243
99%	.705	.652	.045	.211	-5.1E-7	.142	.193

[1] Correlation between predicted and true sire TA's in the simulated data.
[2] Means and standard deviations of actual transmitting abilities (average of five replicates).
[3] Means and standard deviations of estimated transmitting abilities (average of five replicates).
[4] Response to selection of the highest 20%; computed using equation [5], (i = 1.4).

DISCUSSION

This study investigated important properties of threshold model methods for genetic prediction. The validity of TM was confirmed by using computer simulation. Sire evaluations by threshold model prediction methods, performed best with high sire variance, with multiple categorization of outcomes, and with less extreme incidence of categories. These results indicate that TM is expected to give relatively poor predictions and consequently low response to selection with traits of low h^2, of low number of categories, or of rare incidence of any of the categories in the response variable. Some traits such as disease resistance and livability, that are measured on a binary scale, especially if one of the categories has extremely low (or high) incidence could not be accurately predicted by TM. To overcome this difficulty the number of categories in the response variable should be increased unless it is impossible or prohibited practically. Another possibility could be increasing the number of progeny per sire.

Traits such as calving ease and type traits in beef cattle are usually measured on a polychotomous scale and do not have very extreme categories. These traits would be more accurately predicted by TM procedure.

These results do not conflict with previous results reported in (6) where a superiority of TM performance compared with BLUP's as sire variance decreased, number of categories decreased, and incidence of categories became more extreme. This does not necessarily mean an improvement of TM performance, however, it may be due to more deterioration in BLUP performance than TM's as sire variance decreased, number of categories decreased, and incidence of categories became more extreme. This indicates that with some extreme cases TM is not as good as with normal situations, however, it is still better than BLUP.

Eventually, this study reveals the need for more accurate sire evaluation methods for traits of low heritability especially binary traits and traits of extreme incidence in one or several categories of the response variable. The need for more accurate methods could be more justifiable if one notices that this study was conducted using a simple model and balanced data; with more complex models and unbalanced data the accuracy of prediction of TM is expected to deteriorate further.

REFERENCES

1 Gianola, D. 1982. Theory and analysis of threshold characters. J. Anim. Sci. 54:1079.

2 Gianola, D., and J. L. Foulley. 1983. Sire evaluation for ordered categorical data with a threshold model. Genet. Sel. Evol. 15:201.

3 Gilmour, A. R., R. D. Anderson, and, A. L. Rae. 1985. The analysis of binomial data by a generalized linear mixed model. Biometrika 72:593.

4 Harville, D. A., and R. W. Mee. 1984. A mixed-model procedure for analyzing ordered categorical data. Biometrics 40:393.

5 Henderson, C. R. 1973. Sire evaluation and genetic trends. Pages 10-41 in Proc. Anim. Breeding Genet. Symp. in Honor of Dr. Jay L. Lush. Am. Soc. Anim. Sci., Champaign, IL.

6 Jensen, J. 1986. Sire evaluation for type traits with linear and non-linear procedures. Livest. Prod. Sci. 15:165.

7 Meijering, A., and D. Gianola. 1985. Linear versus nonlinear methods of sire evaluation for categorical traits: a simulation study. Genet. Sel. Evol. 17:115.

8 Ronningen, K. 1974. Monte Carlo simulation of statistical-biological models which are of interest in animal breeding. Acta Agric. Scand. 24:135

9 Thompson, R. 1979. Sire evaluation. Biometrics 35:339.

APPENDIX A. COMPUTER PROGRAMS

```
/*****************************************************************************
*TMA1 -- Program to build the matrices: Q, L, and R, of the left hand side *
*        and the vectors:  e, and v of the right hand side of threshold.   *
*        model equations. TMA1 is uesd for first round.                    *
*Author:                                                                    *
*       Gamal A. Abdel-Azim --  October, 1995                               *
*                                                                          *
*Usage:                                                                     *
*      TMA1 <file>  Run the program by typing TMA1 then a file contains    *
*                   vector of categorical observation for the whole data   *
*                                                                          *
*"utility.h": contains the subroutines directly responsible for            *
*             building the required matrices and vectors.                  *
*                                                                          *
*[1]"matrix.h":  contains the subroutins responsible for allocating and     *
*             freeing memory for int, float, or double vectors or matrices *
*                                                                          *
*"def.h": contains common list of definitions                              *
*****************************************************************************/

#include <stdio.h>
#include <stdlib.h>
#include "def.h"
#include "matrix.h"
#include "utility.h"


main(argc, argv)
     int argc;
     char *argv[];
{
  int i,n, l, thn, k, ifail=1;
  float count=0.0;
  float *tc;
  int *tvec;
  double *t, *ndft, *d, *D, *cdft;

  double s15abf_();

  FILE *fpd, *fpp, *fpsol0;

  if(argc != 2) {
     fprintf(stderr, "TMA1: Syntax: TMA1 (file of thresholds)\n");
     exit(-1);
     }

  if ((fpd = fopen(argv[1], "r")) == NULL)
    tmerror("Cannot open data file");

  if((fpp=fopen("param", "w")) == NULL)
    tmerror("Error: open file for parameters");

  if((fpsol0=fopen("sol0", "w")) == NULL)
    tmerror("Error: open file for initial solutions");

  n = records(fpd);
  tvec = ivector(1, n);


  while (!feof(fpd)) {
    for (i=1; i<=n; i++)
      fscanf(fpd, "%d", &tvec[i]);}

  thn = max(tvec, n);
  fprintf(fpp, "n = %d\n", n);
```

[1] the subroutines responsible for allocating and freeing memory for a vector or matix may be obtained from Press et. al. (1992).

```
  fprintf(fpp, "max = %d\n", thn);
  tc = vector(1, thn);
  t = dvector(1, thn-1);
  ndft = dvector(1, thn-1);
  d    = dvector(1, thn);
  D    = dvector(1, thn);
  cdft = dvector(1, thn-1);


  for (l=1; l< thn; l++) {
    for(i=1; i<=n; i++) {
      if(tvec[i] == l)
        k = 1.0; else k = 0.0;
      count += k;}
    tc[l] = count; /* enumerate number of individuals of each category */

  t[l] = cdfa(tc[l]/n); /* cdfa() receives area and returns limit on the X axis */
  fprintf(fpsol0, "%e\n", t[l]-t[1]); /* output initial bound. pt's sol'ns */

  ndft[l]=ndf(t[l]);   }

  fprintf(fpsol0, "%e\n", -t[1]);

  d[1] = -ndft[1];  /* Compute delta1 */

  for(i=2; i<thn; i++)
    d[i] = ndft[i-1] - ndft[i]; /*Get all delta's except the first and last*/

  d[thn] = ndft[thn-1]; /* Get the last delta */

  for(i=1; i<thn; i++)
    {
      cdft[i] = s15abf_(&t[i], &ifail);    /* compute cumulative denseties from 1 to (m-1) */
      fprintf(fpp,"cdft[%d]=%e\n", i,  cdft[i]);
    }

  D[1] = cdft[1];
  fprintf(fpp, "D[1]=%e\n", D[1]);

  for(i=2; i<thn; i++)
    {
      D[i] = cdft[i] - cdft[i-1];
      fprintf(fpp, "D[%d]=%e\n", i, D[i]);
    }

  D[thn] = 1.0 - cdft[thn-1];

  fprintf(fpp, "D[%d]=%e\n", thn, D[thn]);


  buildR(D, d, thn, n); fprintf(fpp, "File out.R has been produced for matrix R\n");
  build_eps(tvec, d, D, n, thn); fprintf(fpp, "File out.eps has been produced for the vector
eps of the RHS\n");

  if(thn > 2)
    {
      buildL(ndft, D, d, thn, n);  fprintf(fpp, "File out.L has been produced for matrix
L\n");
      buildQ(ndft, D, thn, n); fprintf(fpp, "File out.Q has been produced for matrix Q\n");
      build_v(tvec, thn, n, ndft, D); fprintf(fpp, "File out.v has been produced for vector v
of the RHS\n");
    }

  fclose(fpd);
  fclose(fpp);
  free_ivector(tvec, 1, n);
  free_dvector(t, 1, thn-1);
  free_dvector(ndft, 1, thn-1);
  free_vector(tc, 1, thn);
  free_dvector(d, 1, thn);
  free_dvector(D, 1, thn);
  free_dvector(cdft, 1, thn-1);
```

```
}

/*****************************************************************************************/
/** END OF TMA1, file.c **/


/*************************************************************
 * -- "utility.h" --                                         *
 *                                                           *
 * This is file "utility.h". It contains utility subroutines *
 * for TMA1. Used in the first round!                        *
 *************************************************************/

#include <math.h>
#include <stdio.h>
#include <stddef.h>

double ndf(double x)
     /*-----------------------------------------
       evaluate the normal distribution function
       at point x(x is the dependent variable)
       -----------------------------------------*/

{
  return(0.3989422804014327*exp(-(x*x)/2.0));
}
/**END OF ndf()*******************************************************/


double cdfa(double area)
     /*-------------------------------------------------
       This function receives area under the normal curve
       from -infinity to unknown point and returns the point
       -------------------------------------------------*/

{
  double sqrt(), fabs();
  double t,zp,  c0=2.515517, c1=.802853, c2=.010328, d1=1.432788, d2=.189269, d3=.001308;
  if(area == 0.5) return(0.0);
  else if(area < 0.5) { area = 0.5 + fabs(area-0.5);
                        t=sqrt(-2*log(1-area));
                        zp = t-(c0+c1*t+c2*(t*t))/(1+d1*t+d2*(t*t)+d3*(t*t*t));
                        return(-zp);}

  else if (area > 0.5){
    t=sqrt(-2*log(1-area));
    zp = t-(c0+c1*t+c2*(t*t))/(1+d1*t+d2*(t*t)+d3*(t*t*t));
    return(zp);}
}
/**END OF cdfa()******************************************************/


int records(fpd)
     /*-------------------------------------------------------------
       This function counts records in a file and rewinds the file
       -------------------------------------------------------------*/

     FILE *fpd;

     /* records() gets a file pointer */
{
  char line[80];
  int  count=0;

  while (fgets(line, 80, fpd) != NULL) count++;

  rewind(fpd);

  return(count);
}
/**END OF records()*****************************************************/
```

```
int max(t, n)
     /*-------------------------------------------------------------------
       max() receives a one dim'l array, t, along with its length n. It returns
       an int value, the max value found in the array
       -------------------------------------------------------------------*/

     int *t;
     int n;

{
  int i, Max=0;
  for(i=1; i<= n; i++)
    {
      if(Max < t[i])
       Max = t[i];
    }
  return (Max);
}

/**END OF max()*************************************************************/

void buildL(double *ndft,double *D, double *d, int m, int n)

     /*-----------------------------------------------------------
       buildL() builds the matrix L and outputs it into file out.L
       -----------------------------------------------------------*/

{
  FILE *fpL;
  int i, j;

  if ((fpL = fopen("out.L", "w")) == NULL)
    tmerror("Cannot open out.L\n");

  if (m == 3)
    {
      double el;

      el = ndft[2]*((d[2]/D[2])-(d[3]/D[3]));

      for(i=1; i<=n; i++)
        fprintf(fpL, "%e\n", el);
    }

  else if (m > 3)
    {
      double *ael;
      ael = dvector(1, m-2);

      for(i=2; i<m; i++)
        ael[i-1] = ndft[i]*((d[i]/D[i])-(d[i+1]/D[i+1]));

      for(i=1; i<=n; i++)
        {
          for(j=1; j<=m-2; j++)
            fprintf(fpL, "%e  ", ael[j]);
          fprintf(fpL, "\n");
        }
      free_dvector(ael, 1, m-2);

    }

  fclose(fpL);
}
/**END OF buildL()**********************************************************/

void buildQ(double *ndft, double *D, int m, int n)

     /*-----------------------------------------------------------
       buildQ() builds the matrix Q and outputs it into file out.Q
       -----------------------------------------------------------*/
```

```
{
  FILE *fpQ;
  int i, j;

  if(( fpQ = fopen("out.Q", "w")) == NULL)
    tmerror("Cannot open out.Q\n");

  if(m == 3)
    {

      double eq;
      eq = n*square(ndft[2])*((1/D[2])+(1/D[3]));
      fprintf(fpQ, "SPARSE 1 1 1\n");
      fprintf(fpQ, "1 1 %f\n", eq);
    }

  else if(m == 4)
    {
      double fdeq, sdeq, offdeq;
      fdeq = n*square(ndft[2])*(1/D[3]+1/D[3]);
      sdeq = n*square(ndft[3])*(1/D[3]+1/D[4]);
      offdeq = -n*(ndft[2]*ndft[3])/D[3];

      fprintf(fpQ, "SPARSE 2 2 4\n");
      fprintf(fpQ, "1 1 %e\n", fdeq);
      fprintf(fpQ, "1 2 %e\n", offdeq);
      fprintf(fpQ, "2 1 %e\n", offdeq);
      fprintf(fpQ, "2 2 %e\n", sdeq);

    }

  else
    {
      float **Q;
      Q = matrix(1, m-2, 1, m-2);

      for(i=2; i<m; i++)
       Q[i-1][i-1] = n*square(ndft[i])*(1/D[i]+1/D[i+1]);

      for(i=2; i<m-1; i++)
       Q[i-1][i]=Q[i][i-1]= -n*(ndft[i]*ndft[i+1])/D[i+1];


      for(i=1; i < m-1; i++)
        {
          for(j=1; j < m-1; j++)
            fprintf(fpQ, "%e  ", Q[i][j]);
          fprintf(fpQ, "\n");
        }
      free_matrix(Q, 1, m-2, 1, m-2);
    }
  fclose(fpQ);
}

/**END OF buildQ()*************************************************/


void buildR(double *D, double *d, int m, int n)

     /*-----------------------------------------------------------
       buildR() builds the matrix R and outputs it to file out.R
       -----------------------------------------------------------*/

{
  double der = 0.000;
  int i, j;
  FILE *fpR;

  if((fpR = fopen("out.R", "w")) == NULL)
    tmerror("Cannot open out.R\n");

  for(i=1; i<= m; i++)
```

```
    der += square(d[i])/D[i];

  fprintf(fpR, "DIAGONAL %d %d %d\n", n, n, n);

  for(i=1; i<=n; i++)
    fprintf(fpR, "%e\n", der);
  fclose(fpR);
}
/**END OF buildR()***********************************************/

void build_v(int *tvec, int m, int n, double *ndft, double *D)

     /*-------------------------------------------------------
       build_v() receives a pointer to the whole vector of
       observations (tvec), number of categories (m), number of
       records (n), outputs the vector v into file out.v!
       -------------------------------------------------------*/

{
  int k;
  register int i, l;
  int *cat_count; /* ptr to a vector to carry counts for each category */
  FILE *fp_v;

  if((fp_v=fopen("out.v", "w")) == NULL)
    tmerror("Cannot open out.v\n");

  cat_count = ivector(1, m);

  for(i=1; i<=m; i++) cat_count[i] = 0; /* initialize - required - */

  /* next portion enumerate individuals in each category */

  for(i=1; i<=n; i++)  {
    k = tvec[i];
    cat_count[k]++; }


  if(m == 3)
    {
      double ev; /* the only entry of 1x1 v vector */

      ev = cat_count[2]*(ndft[2]/D[2]) - cat_count[3]*(ndft[2]/D[3]);
      fprintf(fp_v, "SPARSE 1 1 1\n");
      fprintf(fp_v, "1 1 %E\n", ev);
    }

  if(m > 3)
    {
      double ev; /* ev differs according to its location in v */

      for(i=2; i<m; i++)
        {
          ev = cat_count[i]*(ndft[i]/D[i]) - cat_count[i+1]*(ndft[i]/D[i+1]);
          fprintf(fp_v, "%E\n", ev);
        }
    }

  free_ivector(cat_count, 1, m);
  fclose(fp_v);

}

/**END OF build_v*************************************************/


void build_eps(int *tvec, double *d, double *D, int n, int m)

     /*-----------------------------------------------
       build_eps() builds the vector Epsilon of the RHS
       and outputs it to file out.eps
       -----------------------------------------------*/
```

```
{
  int i, j;
  FILE *fp_eps;

  if((fp_eps=fopen("out.eps", "w")) == NULL)
    tmerror("Cannot open file out.eps\n");

  for(i=1; i<=n; i++) {
    for(j=1; j<=m; j++) {
      if(tvec[i] == j) {
        fprintf(fp_eps, "%e\n", d[j]/D[j]);
        break;} } }
  fclose(fp_eps);
}
/**END OF build_eps()*************************************************/

/**END OF "utility.h"**/
```

```
/*************************************************************
 * -- def.h --                                                *
 *                                                           *
 * This is file "def.h", it contains some common definitions *
 *************************************************************/

#define SIGN(a,b) ((b) >= 0.0 ? fabs(a) : -fabs(a))

static float maxarg1, maxarg2;
#define FMAX(a,b) (maxarg1=(a), maxarg2=(b), (maxarg1) > (maxarg2) ? (maxarg1) : (maxarg2))

static int iminarg1, iminarg2;
#define IMIN(a,b) (iminarg1=(a), iminarg2=(b), (iminarg1) < (iminarg2) ? (iminarg1) :
(iminarg2))

float pythag(float a, float b);

static float sqrarg;
#define SQR(a) ((sqrarg=(a)) == 0.0 ? 0.0 : sqrarg*sqrarg)
#define square(x) x*x

/*************************************************************************************************
******/
/** END OF def.h **/
```

```
/*********************************************************************
*TMA2 -- Program to build the matrices: Q, L, and R of the left hand*
*        side; and the vectors: e, and v of the right hand side of  *
*        the threshold model equations. TMA2 is for any round after *
*        the first one !                                            *
*                                                                   *
*Author:                                                            *
*       Gamal A. Abdel-Azim -- May, 1996                            *
*                                                                   *
*Usage:                                                             *
*      TMA2 <fixed> <random> <former solutions> <data vector>       *
*                                                                   *
*      <fixed> : is a file contains the matrix of the fixed effects *
*                stored in SPARSE.                                  *
*      <random>: file contains the matrix of the random effects     *
*                stored in SPARSE.                                  *
*      <former solutions>: file contains all solutions from last    *
*                          round stored in SPARSE.                  *
*      <data vector>: file contains vector of categorical response  *
*                     stored in default storage.                    *
*                                                                   *
*"PHI.h": subroutines to compute normal probability and integrals   *
*                                                                   *
*"R2util.h": contains the subroutines directly responsible for      *
*            building the required matrices and vectors.            *
*                                                                   *
*"matrix.h": subroutines for allocating and freeing memory for int, *
*            float, or double vectors or matrices.                  *
*                                                                   *
*"def.h": Common difinitions.                                       *
*********************************************************************/


#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include "def.h"
#include "matrix.h"
#include "utility.h"
#include "PHI.h"
#include "R2util.h"


/*=================================================
  external variables
  n: total number, nxi: m-2, *xi: boundary pts vector
  *mu: vector for the values -xa - zb
  =================================================*/

int n, nxi;
double *xi, *mu;

main(argc, argv)
     int argc;
     char *argv[];
{

  register int i;
  int nx, nz, t, na, nb, nsol, row, k, tn, r, c, dum;
  float temp;
  char str[10];

  int *y;
  double *sol, *alpha, *beta, *x_alpha, *z_beta;


  FILE *fpx, *fpz, *fpsol, *fpd;

  if(argc != 5) {
    printf("TMA2: Syntax: TMA2 X Z SOL Y\n");
```

```
  exit(-1);
}

/* OPEN FILES FOR X Z SOLUTIONS Y*/
/*=======================================*/
if ((fpx = fopen(argv[1], "r")) == NULL)
  tmerror("Cannot open fixed-effects file");

if ((fpz = fopen(argv[2], "r")) == NULL)
  tmerror("Cannot open random-effects file");

if ((fpsol = fopen(argv[3], "r")) == NULL)
  tmerror("Cannot open solutions file");

if ((fpd = fopen(argv[4], "r")) == NULL)
  tmerror("Cannot open data file");


/* Get the first row in the fixed and random effects files
   If the length of X (read in tn) != the length of Z
   (read in n)...ERROR and EXIT
   ========================================================*/
fscanf(fpx, "%s %d %d %d", str, &tn, &na, &nx);

fscanf(fpz, "%s %d %d %d", str, &n, &nb, &nz);

if(n != tn)
  tmerror("NUMBER OF INDIVIDUALS IS DIFFERENT IN FIXED AND RANDOM EFFECTS MATRICES");


/* Read first row in solutions file
   ================================*/
fscanf(fpsol, "%s %d %d %d", str, &nsol, &tn, &t);

/* Make an int vector for observations
   ===================================*/
y = ivector(1, n);

/* Make double vectors for solutions, alpha, and beta.
   Check if nxi >= 1 and make a double vector for xi
   if aplicable
   ================================================*/
sol = dvector(1, nsol);
nxi = nsol - na - nb;

if (nxi > 0)
  xi = dvector(1, nxi);

if (nxi < 0)
  tmerror("Negative number of boundary point solutions");

alpha = dvector(1, na);
beta = dvector(1, nb);

/* Make double vectors to store the product of the ith row
   in X with alpha "x_alpha" and the product of the ith row
   in Z with beta "z_beta". mu is another vector to store
   - x_alpha[] - z_beta[]
   ======================================================*/
x_alpha = dvector(1, n);
z_beta = dvector(1, n);
mu = dvector(1, n);

/* Read data vector y
   ==================*/
for(i=1; i<=n; i++)
  fscanf(fpd, "%d", &y[i]);


/* READ solutions and recover them from SPARSE to normal
   take care of the zero elements. SPARSE stored sol'ns
   are not double. They should be read as float into temp first
```

```
   ============================================================*/
  for(i=1; i<=nsol; i++) {
    fscanf(fpsol, "%d %d %f", &r, &c, &temp);
    if(r == i) {
      sol[r] = (double) temp;
    }
  }


  /* Divide the solutions vector into xi (if applicable), alpha and beta.
     Get red of sol[] to free some memory.
     ====================================================================*/
  if (nxi > 0)
    for(i=1; i<=nxi; i++)
      xi[i] = sol[i];


  for(i=1; i<=na; i++)
    alpha[i] = sol[nxi+i];

  for(i=1; i<=nb; i++)
    beta[i] = sol[nxi+na+i];

  free_dvector(sol, 1, nsol);


  /* READ Fixed and Random effects and get X*alpha and Z*beta
     ========================================================*/
  for(i=1; i<=nx; i++) {
    fscanf(fpx, "%d %d %d", &r, &c, &dum);
    x_alpha[r] += alpha[c]; }


  for(i=1; i<=nz; i++) {
    fscanf(fpz, "%d %d %d", &r, &c, &dum);
    z_beta[r] += beta[c];
  }

  /* Add -(x_alpha[] - z_beta[]) into mu[]
     Get red of x_beta[] and z_beta
     ======================================*/
  for(i=1; i<=n; i++)
    mu[i] = - x_alpha[i] - z_beta[i];

  free_dvector(x_alpha, 1, n);
  free_dvector(z_beta, 1, n);


  if(nxi > 0) {
    buildL2(n, nxi);
    buildQ2(n, nxi);
    buildv2(n, nxi, y);
  }
  buildR2(n, nxi);
  builde2(n, nxi, y);

  fclose(fpx);
  fclose(fpz);
  fclose(fpsol);
  fclose(fpd);

  free_ivector(y, 1, n);
  free_dvector(x_alpha, 1, n);
  free_dvector(z_beta, 1, n);
  free_dvector(mu, 1, n);
  free_dvector(alpha, 1, na);
  free_dvector(beta, 1, nb);
  if(nxi > 0.0) free_dvector(xi, 1, nxi);
}
/*************************************************************/
/** END OF TMA1 **/
```

```
/*****************************************************************************
* -- PHI.h --                                                                 *
*                                                                             *
* phi() returns NDF(i, k) and CDF(i, k) for row i and solutions of the kth    *
* boundary point. It returns NDF if o = 1, and CDF if o = 2. If o = 3, phi()  *
* will return delta, (delta = NDF(i, k-1) - NDF(i, k)). Finally if o = 4,     *
* phi() returns DELTA, (DELTA = CDF(i, k) - CDF(i,k-1)).                      *
*****************************************************************************/

extern int n, nxi;
extern double *xi, *mu;

double phi(int row, int k, int o)
{
  double value;
  double NDF, CDF, delta, DELTA;
  int t, ifail= 1;
  register int i;
  double s15abf_();

  /*================================================================
    THE BOUNDARY POINT NUMBER 0 = -INFINITY; THE VALUE OF -INFIN.
     IN THE NDF = 0. SAME FOR CDF.
   ================================================================*/
  if (k == 0)
    {
    NDF = 0.0000;
    CDF = 0.0000;
  }

  /*================================================================
    IF K = (nxi+2) = LAST CATEGORY. THE NDF = 0.000 AND THE CDF
    = 1.000
    ================================================================*/

  else if (k == (nxi+2))
    {
    NDF = 0.0000E+00;
    CDF = 1.0000E+00;
    value = (xi[nxi] + mu[row]);
    delta = ndf(value);
    DELTA = 1 - s15abf_(&value, &ifail);
  }

  /*============================================================
    THE BONDARY POINT NO. 1 IN THE STAND. THRESHOLD MODEL = 0
   ============================================================*/
 else if (k == 1)
    {
      value = mu[row];
      NDF= ndf(value);
      CDF= s15abf_(&value, &ifail);
      delta = -NDF;
      DELTA = CDF;
    }

  /*============================================================
    THERE ARE ONLY M-2 BOUNDARY POINT SOL'NS SOLUTIONS. THEY ARE
    IN THE INTERVAL[2, M-2] OR [2, nxi+1].
    ============================================================*/
  else if(k <= (nxi+1) && k >1)
    {
      value = (xi[k-1] + mu[row]);
      NDF = ndf(value);
      CDF = s15abf_(&value, &ifail);

      value = (xi[k-2] + mu[row]);
      delta = ndf(value) - NDF;
      DELTA = CDF - s15abf_(&value, &ifail);
    }
```

```
  /*=========================
    ANYTHING ELSE IS INVALID
    ========================*/
  else
    {
      tmerror("invalid category\n");
      exit(-1);
    }

  if (o == 1)
    return(NDF);

  if (o == 2)
    return(CDF);

  if (o == 3)
    return(delta);

  if (o == 4)
    return(DELTA);
}
/*****************************************************************/

/** END OF PHI.h **/
```

```
/*****************************************************************
*-- R2util.h --                                                  *
*                                                               *
*This is file "R2util.h". Contains utility subroutines for TMA2*
*used in building threshold model equations for second and      *
*subsequent rounds                                             *
*****************************************************************/

void buildQ2(int n, int nxi)

     /*===============================================================
       buildQ() builds the matrix Q for second and further iterates and
       outputs it into the file out.Q
       ===============================================================*/

{
  FILE *fpQ;
  int i, j, k;
  register int row;

  double phi_ik, phi_ikp1, DELTA_ik, DELTA_ikp1;

  if ((fpQ = fopen("out.Q", "w")) == NULL)
    tmerror("Cannot open out.Q\n");

  if (nxi == 1)
    {
      double eq = 0.000;

      for(row=1; row<=n; row++)
        {
          k = 2;   /* The only solution of the second boundary point 3 cat's are there */
          phi_ik= phi(row, k, 1);

          DELTA_ik = phi(row, k, 4);
          DELTA_ikp1 = phi(row, k+1, 4);

          eq += (phi_ik*phi_ik)*(1/DELTA_ik + 1/DELTA_ikp1);
        }
      fprintf(fpQ, "SPARSE 1 1 1\n");
      fprintf(fpQ, "1 1 %E\n", eq);
    }
  else if (nxi > 1)
    {
      double **mq;
      double deq = 0.000, offdeq = 0.000;
      mq = dmatrix(1, nxi, 1, nxi);

      /*=========================
        set elements of Q to 0.0!!
        =========================*/
      for(i=1; i<=nxi; i++)
        for(j=1; j<=nxi; j++)
          mq[i][j] == 0.0;


      /*===========================================================
        It's important for k to begin from 2. Notice that k itself
        is sent to phi for computing phi's and DELTA's.
        ===========================================================*/

      for(k=2; k<(nxi+2); k++)
        {
          for(row=1; row<=n; row++)
            {
              phi_ik= 10000*phi(row, k, 1);
              phi_ikp1= 10000*phi(row, k+1, 1);

              DELTA_ik = 10000*phi(row, k, 4); /* Magnify to avoid division by 0.0 */
              DELTA_ikp1 = 10000*phi(row, k+1, 4);
```

```
            deq += 0.0001*(phi_ik*phi_ik)*(1/DELTA_ik + 1/DELTA_ikp1);
            if(k<nxi+1)
              offdeq += 0.0001*(phi_ik*phi_ikp1)/DELTA_ikp1;

          }
        mq[k-1][k-1] = deq;
        if(k<nxi+1) {
          mq[k-1][k] = -offdeq;
          mq[k][k-1] = -offdeq;
        }

        deq = 0.000;
        offdeq = 0.000;

      }

    for(i=1; i<=nxi; i++)
      {
        for(j=1; j<=nxi; j++) {
          if (fabs(mq[i][j]) < 1.0e-20) mq[i][j] = 0.00;
          fprintf(fpQ, "%E  ", mq[i][j]);
        }
        fprintf(fpQ, "\n");
      }


    free_dmatrix(mq, 1, nxi, 1, nxi);
  }

  fclose(fpQ);
}

/**END OF buildQ2()*****************************************************************/
/**********************************************************************************/


void buildL2(int n, int nxi)

    /*=================================================================
      buildL() builds the matrix L for second and later iterates and
      outputs it into the file out.L
      ================================================================*/

{
  FILE *fpL;
  int j, k;
  register i, row;

  double phi_ik, delta_ik, delta_ikp1, DELTA_ik, DELTA_ikp1;

  if ((fpL = fopen("out.L", "w")) == NULL)
    tmerror("Cannot open out.L\n");

  if (nxi == 1)
    {
      double *vl;
      vl= dvector(1, n);

      for(row=1; row<=n; row++)
        {
          k = 2;
          phi_ik= phi(row, k, 1);

          delta_ik= 10000*phi(row, k, 3);
          delta_ikp1 = 10000*phi(row, k+1, 3);

          DELTA_ik = 10000*phi(row, k, 4);
          DELTA_ikp1 = 10000*phi(row, k+1, 4);

          /* mult by 10000 to avoid vey small values caused by subtraction */

          vl[row]=phi_ik*10000*((delta_ik/DELTA_ik) - (delta_ikp1/DELTA_ikp1));
```

```
          fprintf(fpL, "%E\n", vl[row]/10000);

        }
      free_dvector(vl, 1, n);
    }
  else if (nxi > 1)
    {
      double **ml;
      ml = dmatrix(1, n, 1, nxi);

      for(k=2; k<(nxi+2); k++)
        for(row=1; row<=n; row++)
          {
            phi_ik= phi(row, k, 1);

            delta_ik= 10000*phi(row, k, 3);
            delta_ikp1= 10000*phi(row, k+1, 3);

            DELTA_ik = 10000*phi(row, k, 4);
            DELTA_ikp1 = 10000*phi(row, k+1, 4);

            ml[row][k-1]=phi_ik*10000*((delta_ik/DELTA_ik) - (delta_ikp1/DELTA_ikp1));

          }

      for(i=1; i<=n; i++)
        {
          for(j=1; j<=nxi; j++)
            fprintf(fpL, "%E  ", ml[i][j]/10000);
          fprintf(fpL, "\n");
        }

      free_dmatrix(ml, 1, n, 1, nxi);
    }

  fclose(fpL);
}

/**END OF buildL2()**************************************************************************/
/*********************************************************************************************/


void buildR2(int n, int nxi)


     /*================================================================
       buildR2() builds the matrix R for second and further iterates and
       outputs it into the file out.R
       ================================================================*/

{
  FILE *fpR;
  int k;
  register int row;

  double delta_ik, DELTA_ik, der = 0.000;

  if ((fpR = fopen("out.R", "w")) == NULL)
    tmerror("Cannot open out.R\n");

  fprintf(fpR, "DIAGONAL %d %d %d\n", n, n, n);

  for(row=1; row<=n; row++) {
    for(k=1; k<=(nxi+2); k++) {
      delta_ik = 10000*phi(row, k, 3);
      DELTA_ik = 10000*phi(row, k, 4);
      der += 0.0001*square(delta_ik)/DELTA_ik;
    }
    fprintf(fpR, "%E\n", der);
    der = 0.000;
  }
```

```
  fclose(fpR);
}


/*END OF buildR2()*************************************************************/
/*****************************************************************************/


void builde2(int n, int nxi, int *y)
     /*================================
       y: vector of categorical observ's
       ================================*/

     /*=================================================================
       builde2() builds eps. vector for second and further iterates and
       outputs it into the file out.eps
       =================================================================*/

{
  FILE *fpe;
  int k;
  register int row;
  double delta_ik, DELTA_ik;

  if((fpe = fopen("out.eps", "w")) == NULL)
    tmerror("Cannot open file for out.eps\n");

  for(row=1; row<=n; row++) {
    k = y[row];
    delta_ik = 10000*phi(row, k, 3);
    DELTA_ik = 10000*phi(row, k, 4); /* Magnify to avoid division by 0.0 */
    fprintf(fpe, "%E\n", delta_ik/DELTA_ik);
  }
  fclose(fpe);
}

/**END OF builde2()************************************************************/
/*****************************************************************************/


void buildv2(int n, int nxi, int *y)

     /*================================
       y: vector of categorical observ's
       ================================*/

     /*=================================================================
       buildv2() builds v vector for second and further iterates and
       outputs it into the file out.v
       =================================================================*/

{
  FILE *fpv;
  int k;
  register int i, row;

  double *set1, *set2;
  double phi_ik, DELTA_ik;

  if ((fpv = fopen("out.v", "w")) == NULL)
    tmerror("Cannot open out.v\n");


  /*=============================================================================
    set1 is a vector contains SUM(i as an element of k){phi_ik/DELTA_ik
    set2 is a vector contains SUM(i as an element of k+1){phi_ik/DELTA_ik+1
    v = set1 - set2!
    =============================================================================*/


  set1 = dvector(1, nxi);
  set2 = dvector(1, nxi);
```

```
  /*=========================
    set set1 and set2 to 0.0
    ========================*/
  for(i=1; i<= nxi; i++) {
    set1[i]=0.0;
    set2[i]=0.0;
  }

  for(row=1; row<=n; row++) {
    k = y[row];  /* Look up the value y[row] once */
    if(k > 1 && k < nxi+2) {

      phi_ik= 10000*phi(row, k, 1);

      DELTA_ik = 10000*phi(row, k, 4);

      set1[k-1] += phi_ik/DELTA_ik;
    }
  }

  for(row=1; row<=n; row++) {
    k = y[row];  /* Look up the value in y[row] once */

    if(k > 2) {

      phi_ik = 10000*phi(row, k-1, 1);
      DELTA_ik = 10000*phi(row, k, 4);
      set2[k-2] += phi_ik/DELTA_ik;
    }
  }

  fprintf(fpv, "SPARSE %d 1 %d\n", nxi, nxi);
  for(k=1; k<=nxi; k++)
    fprintf(fpv, "%d 1 %E\n", k, set1[k] - set2[k]);

  fclose(fpv);
  free_dvector(set1, 1, nxi);
  free_dvector(set2, 1, nxi);
}

/**END OF buildv2()************************************************************/
/*****************************************************************************/

/** END OF R2util.h **/
```

APPENDIX B. SHELL SCRIPT FOR THRESHOLD MODEL ANALYSIS FOR CALVING EASE DATA.

```
#! /bin/tcsh

#####################################################
# Script file for solving threshold model equations#
# using TMA1, TMA2, some UNIX utilities, and some  #
# ABTK tools.                                       #
#                                                   #
# TMA1: builds the threshold matrices for the first#
#       round.                                      #
#                                                   #
# TMA2: builds the threshold matrices for the       #
#       second and subsequent rounds.               #
#                                                   #
# Builds and solves threshold model equations for  #
# sex parity as fixed and HYS and sire as random   #
# for the calving ease data                         #
#####################################################


#============================
# Threshold Model Equations
#============================

#    | Q   L`X   L`Z     | |v|   |v       |
#    |     X`RX  X`RZ    | |a| = |X`e     |
#    | Sym       Z`RZ+D-1| |b|   |Z`e-D-1b|


#=================================================
# Prepare the data, then X, Z, and threshold part!
#=================================================

#-------------------------------------------------
# The data file contains the following effects:   #
#                                                 #
# 1.  sex in the first field of the file.         #
# 2.  parity in the second field.                 #
# 3.  herd-year-season in the third field.        #
# 4   sire in the fourth field.                   #
# 5.  categorical observations in the fifth field #
#-------------------------------------------------

awk '{print $1}' data >! sex
awk '{print $2}' data >! parity
awk '{print $3}' data >! hys
awk '{print $4}' data >! sire
awk '{print $5}' data >! y

#===
# X
#===

mu y >! x_mu
fef -d sex -r y -e sex.list >! x_sex
fef -d parity -r y -e parity.list >! x_parity
d2s x_mu x_mu.s
scat x_mu.s x_sex x_parity X

rm sex parity x_mu x_mu.s x_sex x_parity
#===
# Z
#===

sort -u hys >! hys.list
gen_z -d hys -e hys.list -r y -o z_hys

sort -u sire >! sire.list
gen_z -d sire -e sire.list -r y -o z_sire
```

```
scat z_hys z_sire Z

rm hys sire z_hys z_sire


#=======================================
# ``TMA1`` outputs The Threshold Part:
# out.Q  out.L  out.R  out.eps  out.v
#=======================================

TMA1  y

d2s out.Q outs.Q
#=====================
# Build [Q  L`X  L`Z]
#=====================

mult -tout.L X LtX
mult -tout.L Z LtZ
scat outs.Q LtX LtZ V.EQ
rm outs.Q out.Q LtX LtZ

#=========================
# Build [X`L  X'RX  X'RZ]
#=========================

mult -tX out.R XtR
mult XtR X XtRX
mult XtR Z XtRZ
mult -tX out.L XtL
scat XtL XtRX XtRZ X.EQ
rm XtL XtRX XtRZ XtR

              #==============================
              # Build [Z'L  Z`RX  Z`RZ+D-1]
              #==============================
                     #============
                         #=====
                           #==


mult -tZ out.R ZtR
mult ZtR Z ZtRZ

#=========================
#Construct the matrix D-1,
#where, D-1 = |D1-1 0   |
#             |0    D2-1|
# then, add ZtRZ + D-1
#=========================

#-----------------------------------#
# This data set contains 62758 hys's #
# and 5593 sires. D(inv) is to be    #
# built based on these numbers       #
#-----------------------------------#

ident.abtk 62758 >! I.u
mult "I.u*10.4" I.u D.h

nullm 62758 5593 >! nul.u
scat D.h nul.u D.u

ident.abtk 5593 >! I.l
mult "I.l*25.97" I.l D.s

nullm 5593 62758 >! nul.l
scat nul.l D.s D.l

svcat D.u D.l D
```

```
sadd ZtRZ D ZtRZD


mult -tZ out.L ZtL
mult ZtR X ZtRX
scat ZtL ZtRX ZtRZD Z.EQ

rm I.* D.* nul.* ZtL ZtRX ZtRZ ZtRZD ZtR

#=====
# LHS
#=====

svcat V.EQ X.EQ VX.EQ
svcat VX.EQ Z.EQ LHS
rm *.EQ

#===========
# Build RHS
#===========
d2s out.v outs.v
mult -tX out.eps Xteps
mult -tZ out.eps Zteps
svcat outs.v Xteps vx
svcat vx Zteps RHS
rm vx Zteps Xteps out.*

# Don't need D(inv)*b in the first round. b is a vector of zero's

#=========================================================
# Get an iterative solution for threshold model equations
# Use 100 iterations!
# Change storage mode of beta to SPARSE by d2s
#=========================================================

itgen -m 100 -r RHS -n LHS -b beta
d2s beta beta.s
rm LHS RHS

#=========================================
# add the corrections in beta.s to initial
# values in sol0.s, produced by TMA1
#=========================================

sadd beta.s sol0.s sol.s
echo "DONE FIRST ROUND"

            ##############################################
            # The following is an infinite shell loop    #
            # it may be used to build the equations      #
            # several times until a desired convergence  #
            # is met. You may break the loop by Ctrl+c   #
            # or you may set a conditional expression !  #
            ##############################################
                    ############################
                          ##############
                                ###


######################
# Begin iteration    #
######################

repeat:

TMA2 X Z sol.s y

d2s out.Q outs.Q

mult -tout.L X LtX
mult -tout.L Z LtZ
```

```
scat outs.Q LtX LtZ V.EQ

rm out.Q outs.Q LtX LtZ

mult -tX out.R XtR
mult XtR X XtRX
mult XtR Z XtRZ
mult -tX out.L XtL
scat XtL XtRX XtRZ X.EQ

rm XtR XtRZ XtRX XtL

mult -tZ out.R ZtR
mult ZtR Z ZtRZ

sadd ZtRZ D ZtRZD

mult -tZ out.L ZtL
mult ZtR X ZtRX
scat ZtL ZtRX ZtRZD Z.EQ

rm ZtR ZtRZ ZtRZD ZtL ZtRX

svcat V.EQ X.EQ VX.EQ
svcat VX.EQ Z.EQ LHS

rm *.EQ

d2s out.v outs.v
mult -tX out.eps Xteps
mult -tZ out.eps Zteps
svcat outs.v Xteps vx

################################
# compute D(inv)*b             #
# there are -- 9 -- equations  #
# for the fixed effects and    #
# boundary points              #
################################

mprint sol.s >! sol
awk 'NR > 9 {print $1}' sol >! b
mult D b Db
sadd Zteps "Db*-1" ZtepsD

svcat vx ZtepsD RHS
rm vx Xteps Zteps ZtepsD Db out.*

itgen -m 100 -r RHS -n LHS -b beta
d2s beta beta.s

rm LHS RHS

sadd beta.s sol.s sol.s.temp
mv sol.s.temp sol.s

echo "DONE ANOTHER ROUND"

goto repeat

################################
#### END OF SHELL SCRIPT FILE ###
################################
```

REFERENCES

Berger, P. J., and A. E. Freeman. 1978. Prediction of sire merit for calving difficulty. J. Dairy Sci. 61: 1146.

Berger, P. Jeffrey. 1994. Genetic Prediction for calving ease in the United States: data, models, and use by dairy industry. J. Dairy Sci. 77: 1146.

Bock, R. D. 1975. Multivariate Statistical Methods in Behavioral Research. McGraw-Hill Book Co., New York.

Bulmer, M. G. 1980. The Mathematical Theory of Quantitative Genetics. Clarendon Press, Oxford.

Clutter, A. C., P. J. Berger, and J. M. Mattison. 1989. Threshold-model analysis of dystocia in dairy cattle when progeny information is limited. J. Dairy Sci. 72: 3264.

Curry, D. A. 1991. Using C on the UNIX System. O'Reilly, Sebastopol, CA.

Dempster, E. R., and I. M. Learner. 1950. Heritability of threshold characters. Genetics 35: 212.

Djemali, M., P. J. Berger, and A. E. Freeman. 1987. Ordered categorical sire evaluation for dystocia in Holsteins. J. Dairy Sci. 70: 2374.

Falconer, D. S. 1989. Introduction to Quantitative Genetics. Longman, England.

Gianola, D. 1980. A method of sire evaluation for dichotomies. J. Anim Sci. 51: 1266.

Gianola, D. 1982. Theory and analysis of threshold characters. J. Dairy Sci. 54: 1079.

Gianola, D., and J. L. Foulley. 1983. Sire evaluation for ordered categorical data with a threshold model. Genet. Sel. Evol. 15: 201.

Golden, B. L., W. M. Snelling, and C. H. Mollinckrodt. 1992. Animal Breeders Toolkit: User's Guide and Reference Manual. Colorado State Univ. Agric. Exp. Sta. Tech. Bull. LTB92-2.

Grizzle, J. E., C. F. Starmer, and G. G. Koch. 1969. Analysis of categorical data by linear models. Biometrics 25: 489.

Harville, D. A., and R. W. Mee. 1984. A mixed model procedure for analyzing ordered categorical data. Biometrics 40: 393.

Henderson, C. R. 1973. Sire evaluation and genetic trends. Proc. of Anim. Breeding and Genetics Symp. in honor of Dr. J. L. Lush, Amer. Soc. of Anim. Sci. and Amer. Dairy Sci. Assoc., Champaign, IL.

Henderson, C. R. 1975. Inverse of a matrix of relationships due to sires and maternal grandsires. J. Dairy Sci. 58: 1917.

Kennedy, W. J., and Gentle, J. E. 1980. Statistical Computing. Marcel Decker, NY.

McKelvey, R. D., and Zavoina, W. 1975. A statistical model for the analysis of ordinal level dependent variables. J. Mathematical Sociology 4: 103.

Mee, R. W. 1981. Analysis of ordered responses, assuming an underlying continuous variable. Ph. D. dissertation. Iowa State. University, Ames, Iowa.

Naazie, A., M. Makarechian, and R. T. Berg. 1989. Factors influencing calving difficulty in beef heifers. J. Animal Sci. 67: 3243.

Naazie, A., M. Makarechian, and R. T. Berg. 1991. Genetic, phenotypic, and environmental parameter estimates of calving difficulty, weight, and measures of pelvic size in beef heifers. J. Animal Sci. 69: 4793.

NAG. 1993. NAG Fortran Library, Mark 15. NAG Ltd, Oxford.

Oualline, S. 1993. Practical C Programming. O'Reilly, Sebastopol, CA.

Philipsson, J. 1976a. Studies on calving difficulty, stillbirth and associated factors in Swedish cattle breeds. I. General introduction and breed averages. Acta Agric. Scand. 26: 151.

Philipsson, J. 1976b. Studies on calving difficulty, stillbirth and associated factors in Swedish cattle breeds. I. Effects on non genetic factors. Acta Agric. Scand. 26: 211.

Pollak, E. J., and A. E. Freeman. 1976. Parameter estimation and sire evaluation for dystocia and calf size in Holsteins. J. Dairy Sci. 59: 1817.

Press, W. H., S. A. Teukolsky, W. T. Vetterling, and B. P. Flannery. 1992a. Numerical recipes in C: The Art of Scientific Computing. Univ. of Cambridge Press. NY.

Press, W. H., S. A. Teukolsky, W. T. Vetterling, and B. P. Flannery. 1992b. Numerical recipes: Example Book [C]. Univ. of Cambridge Press. NY.

Schildt H. 1990. C: The Complete Reference. Osborne McGraw-Hill. Berkeley, Calif.

Searle, S. R. 1971. Linear Models. John Wiley and Sons, New York.

Snell, E. J. 1964. A scaling procedure for ordered categorical data. Biometrics 20: 592.

Thompson, R. 1979. Sire evaluation. Biometrics 35: 339.

Tong, A. K. W., J. W. Wilton, and L. R. Schaeffer. 1977. Applications of a scoring procedure and transformation to dairy type classification and beef ease of calving categorical data. Can. J. Anim. Sci. 57: 1.